REDUCING THE SEARCH TIME OF A GENETIC ALGORITHM USING THE IMMIGRATION OPERATOR

NAGW-1333

by

Michael C. Moed, Charles V. Stewart, and Robert B. Kelley

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering
and
Department of Computer Science
Troy, New York 12180-3590

October, 1990

CIRSSE REPORT #69

REDUCING THE SEARCH TIME OF A GENETIC ALGORITHM USING THE IMMIGRATION OPERATOR

Michael C. Moed†, Charles V. Stewart‡ and Robert B. Kelley†

†Center for Intelligent Robotic Systems for Space Exploration (CIRSSE)
Department of Electrical, Computer and Systems Engineering

‡Department of Computer Science

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Abstract

This paper examines the fundamental tradeoff between *exploration* and *exploitation* in a Genetic Algorithm (GA). An *immigration* operator is introduced, that infuses random members into successive GA populations. It is theorized that immigration maintains much of the exploitation of the GA while increasing exploration. To test this theory, a set of functions are designed that often require the GA to perform an excessive number of evaluations to find the global optimum of the function. For these functions, it is shown experimentally that a GA enhanced with immigration (1) reduces the number of trials that require an excessive number of evaluations and (2) decreases the average number of evaluations needed to find the function optimum

I. Introduction

The tradeoff between exploration and exploitation in serial Genetic Algorithms (GA's) for function optimization is a fundamental issue [1]. If a GA is biased towards exploitation, highly fit members are repeatedly selected for recombination. Although this quickly promotes better members, the population can prematurely converge to a local optimum of the function. On the other hand, if a GA is biased towards exploration, large numbers of schemata are sampled which tends to inhibit premature convergence. Unfortunately, excessive exploration results in a large number of function evaluations, and defaults to random search in the worst case. To search effectively and efficiently, a GA must maintain a balance between these two opposing forces.

This study experimentally examines an *immigration* operator which, for certain types of functions, allows increased exploration while maintaining nearly the same level of exploitation for the given population size. Section II provides relevant background material on this topic and develops the motivation for this study. In section III, we describe the immigration operator, its incorporation into the evaluation, selection and recombination cycle of a Genetic Algorithm, and predict the behavior of the "Modified" Genetic algorithm. In section IV, the implementation of two genetic algorithms is described. One algorithm is based on steady state GA's, and the other is based on restarted GA's as described by Goldberg [2]. Also described is the implementation of the two GA's modified with the immigration operator.

To compare the performance of each GA with and without immigration, a suite of test functions is developed. Each function is characterized by different types of local and global optima. The local optima are designed to provide traps that the genetic algorithm must successfully avoid or recover from in order to achieve a global optimum. These functions are defined in section V. Section VI discusses the experiments performed on the test suite and presents each GA experiment in terms of fitness assignment, population sizes, mutation rate and immigration rate. The results of the GA experiments are examined in terms of the number of evaluations required to find the global optimum of each function and are presented in Section VII. It is shown that a GA modified with immigration reduces the average number of evaluations required to find the function optimum over a range of population sizes. It is further shown that the number of trials requiring an excessive number of evaluations is reduced for functions in this set. Section VIII provides conclusions and recommendations for further research.

II. Background and Motivation

Population size is one parameter that directly effects the balance between exploration and exploitation. DeJong [3] noted that increasing the population size improves long-term performance of a GA at the expense of on-line performance for his test suite. In an extensive study, Schaffer et. al. [4] tested GA parameters for an expanded suite of functions and measured on-line performance. The study indicated that functions with many local optima have good on-line performance with larger population sizes. However, the study noted that an excessively large population imposes an increased number of evaluations per generation and produces poor overall performance.

As these studies showed, increasing the population size of a GA supplements the amount of "raw material" available for processing. If the necessary material is present in the initial population, the GA can converge on an optimal solution. If the initial population size is very large, the optimal schemata may very likely be present and the optimal solution will be found.

However, large population size can lead to an inefficient GA. The GA processes the schemata contained in the "raw material" and exploits those schemata that perform well. As the population size grows and the selection "pie" is divided into more slices, the exploitation decreases. A decrease in exploitation means that better schemata propagate at a slower rate. This forces the GA to increase the number of population samples over repeated generations in order to determine the optimal schemata, and slows convergence of the GA.

Goldberg [2] also emphasized the tradeoff between schema processing (exploration) and convergence rate (exploitation) with regard to population size. To combat slow convergence in serial GA's while finding the optimum of a function, Goldberg suggested using small population GA's that are restarted after convergence. The restart procedure consists of keeping only the best individuals of the converged population and replacing other members by randomly generated individuals. Goldberg shows that this procedure maintains a high rate of schema processing, a cost measure developed to examine GA performance. Intuitively, this technique maintains a very high level of exploitation, since small populations rapidly converge to the best schemata present. Exploration is also enhanced by restarting the population with random members after convergence. Therefore, this technique seems to achieve some balance between the two GA forces.

Given that small population GA's may not possess the necessary schemata to find the optimal solution, and large population GA's can be inefficient in schema processing, this study examines the effect of continually replacing the worst members of a GA population

with random members. The technique is called *immigration*. For GA's of the type tested by DeJong [3] and Schaffer et. al. [4], immigration increases the amount of "raw material" available to the GA, and enhances exploration without increasing population size. For Goldberg's restarted GA, immigration allows increased exploration while the population converges, and may prevent quick convergence to a local optimum. This is especially significant if the test function contains local optima that are difficult to escape.

III. A GA with the Immigration Operator

To balance exploration with exploitation, we propose the following algorithm that incorporates the immigration operator into the general structure of a Genetic Algorithm. A preliminary version of this algorithm was originally presented in [5].

The Modified Genetic Algorithm

1. Evaluate each member of the population and assign a fitness value.
2. Replace m current worst members of the population with m randomly generated and evaluated members. (Immigration Operator)
3. Probabilistically select a subset of members based on fitness.
4. Recombine selected members to form children.
5. Replace the worst members of the population with children.
6. Mutate some members to maintain population diversity and perform local search. Mutation is not performed on one copy of the current best member.

With each generation of the GA, random members are immigrated and replace the worst members in the population. It is important to note that the number of random individuals substituted into the population each generation (m, called the *immigration rate*) is small compared to the size of the population. The advantages and tradeoffs of immigration are described below:

1. When a GA is initialized its population of n random members must contain most of the "raw material" required to assemble the optimal string through selection and crossover. With smaller population GA's, the necessary schemata to build the optimal string may not be present in the initial population. If this is the case, the GA must rely on mutation to bring in the necessary schemata, which can be very inefficient,

especially in problems with many local optima. The immigration operator allows the GA to sample many more individuals during search, and more easily acquire the necessary structure to find the optimal string. However, immigration does not increase the size of the population, since random members replace poor performers. Therefore, *immigration allows a size n population to explore the space of a larger population.*

2. Since the actual population size is not increased to accomplish the added exploration, the high performing schemata in the population can propagate at nearly the same rate as a GA without immigration. In contrast, if the population size is increased to enhance exploration, the schemata propagate more slowly, due to decreased selection pressure on good schemata. Therefore, immigration increases exploration while maintaining selection pressure (exploitation).

3. When a GA operates on a deceptive function, low-order schemata that are present in the optimal string have poor average fitness values. Individuals containing these schemata perform poorly, and are replaced in the population during the selection and recombination process. Immigration provides the GA with repeated opportunities to acquire optimal building blocks, even after they have been discarded.

4. The inclusion of an immigration operator does force a tradeoff in the GA. Immigration exchanges poor performers with random members. Each random member must be evaluated, which may increase the number of evaluations required to find the function optimum. However, immigrants can also bring missing structure to the population which should reduce the number of evaluations required to find the optimum. Therefore, the tradeoff of increased evaluations vs. increased structure must be examined experimentally to determine the applicability of immigration in a GA.

One way to look at a GA modified with the immigration operator is as a GA with a large "virtual" population that maintains much of the selection pressure of a smaller population.

IV. The Implementation of Two Genetic Algorithms

In this study, two different GA's were implemented, each with and without immigration. The first algorithm is a steady state GA. Each iteration of the algorithm we used is described as follows:

Steady State GA

1. Evaluate each new member of the population and assign a fitness value.
2. Probabilistically select two members from the population based on fitness. These members are parents.
3. Perform one-point crossover on the parents at a random string position to form two children.
4. Probabilistically perform mutation on the children.
5. Replace the two worst members of the population with the children.
6. Probabilistically perform mutation on the rest of the population (optional).

The algorithm is modified to include immigration by the addition of the following step:

1.5. Generate and evaluate m random members and replace the m worst members of the population with the m random members.

The second algorithm is based on Goldberg's restarted GA. Given a population of size n, each iteration of the algorithm proceeded as follows:

Restarted GA

1. Evaluate each member of the population and assign a fitness value.
2. Compute the bitwise convergence of the population.
3. If the convergence is greater than a given threshold, replace all but the best two members of the population with randomly generated and evaluated members.
4. Probabilistically select n - 2 members from the population based on fitness.
5. Randomly order the n - 2 selected individuals and form pairs. These pairs are parents.
6. Perform one-point crossover on each set of parents at a random string position to form children.
7. Replace the n - 2 worst members of the population with the children.

8. Probabilistically perform mutation on the children.

For selection, the restarted GA used Stochastic Universal Sampling as described by Baker [6].

The algorithm is modified to include immigration by the addition of the following step:

1.5. Generate and evaluate m random members and replace the m worst members of the population with the m random members.

Only n -2 members are selected each generation to insure survival of the best two performing members.

V. Test Suite of Functions

As stated earlier, immigration imposes a tradeoff in a GA. The added structure introduced by the random members occurs at the cost of evaluating each random member immigrated. Given this tradeoff, the type of functions where immigration should achieve a favorable balance between these factors and increase performance of the steady state GA are those functions with local optima that are difficult to avoid or escape. Functions of this nature require the steady state GA to be more circumspect while converging, and therefore may require sampling more structure. Immigration introduces the needed structure, which may have been discarded from the population. By increasing the structure available to the GA, immigration should reduce the chance of converging at a local optimum, and thereby reduce the overall number of function evaluations by the steady state GA.

On the other hand, on functions that are unimodal, a steady state GA with immigration should perform poorly. A steady state GA operating on a unimodal function has a reduced chance of losing the structure necessary to find the global optimum. Adding the immigration operator introduces redundant structure at the cost of function evaluations. In this case, the tradeoff between added structure vs. added evaluations does not achieve a favorable balance, since the added structure would already be in the population. Therefore, a steady state GA with immigration should increase the number of function evaluations required to find the global optimum of a unimodal function.

It is difficult to predict the effect of immigration on a restarted GA. Like the steady state GA, immigration should allow a small population GA to sample more structure, and help prevent the GA from settling into a local optimum from which it is difficult to escape. This should reduce the number of function evaluations required by the GA for multimodal functions.

For unimodal functions, it is reasonable to believe that the small population must converge a number of times before the global optimum of the function is found. Each convergence imports a host of new random members. It is difficult to predict whether importing random members after convergence is more efficient than immigrating random members during convergence, since both techniques maintain a high rate of schema processing.

The experimental suite consists of a set of six functions that have different types of local and global optima. The suite was created to examine the ability of a GA to escape or avoid difficult local optima. This is reflected in the number of function evaluations required to find the global optimum of the function. By using a suite of this nature, one can determine the type of problems that prove difficult for a GA to solve, and show how the immigration operator affects performance.

Each of the functions is defined on a 20 bit binary string. The optimum cost value for each function is 0.0, which is the minimum value of the function. The functions are described as follows:

1. F1 (ODDEVEN): The purpose of this function is determine how well a GA can combine low-order, high-performing schemata into a structure where good local performance may lead to poor global fitness.

   A sliding window of length 4 is moved one bit at a time over a twenty bit member. The maximum cost is assigned 17.0. Each time the pattern 0101 or 1010 appeared in the window, 1.0 is subtracted from the cost. A pattern of alternating 1's and 0's, 01010101010101010101 or its complement produces the minimum cost of 0.0. A string of all 1's or all 0's has the maximum cost of 17.0.

   For example, the string 01010111111111111111 has a functional value of 17.0 - 3.0 = 14.0 since there are three 4 bit strings of alternating patterns. The first one starts at position 0 and is 0101, the second begins at position 1 and is 1010 and the third begins at position 2 and is 0101.

   This function contains local minima (optima) that may trap the GA. Consider the member 01010101011010101010. This member has a cost of 17.0 - 14.0 = 3.0 which would indicate that it is a near optimal member. However, the member must actually invert the values of 10 consecutive bits in order to achieve the the optimal configuration, which is a large Hamming distance. Such large Hamming distance disturbances are difficult to create through the mutation operator without destroying the good structure in the population. Therefore, if the population converged around this pattern or a similar one, the GA would be trapped in a local optimum. The example

demonstrates that the combination of short-order, high-performing building blocks may produce a string that is far from the global optimum.

2. F2 (DECEPT1): The purpose of this function is to determine the performance of a GA on a difficult, two bit deceptive problem.

   The 20 bit member represents four non-overlapping fields, each of length 5 bits. The maximum cost is assigned 28.0. For each five bit field, 1.2 is subtracted from the maximum cost for each bit in the field that is a 1. However, if all five bits in the field are 0, 7.0 is subtracted from the maximum cost. Therefore, a field of five 1 bits subtracts 6.0 from the maximum cost, so the member 11111111111111111111 has a cost of 4.0. The minimum cost member is all 0's, and has a cost of 0.0.

   For example, the member 00000111111010100001 subtracts 7.0 for the maximum cost for bits 0-4, 6.0 for bits 5-9, 3.6 for bits 10-14 and 1.2 for bits 15-20, for a total cost of 28.0 - 17.8 = 10.2

   This function falls within the class of GA-hard problems [1], since it contains low-order deceptive schemata. These indicate that the function minimum is a string of all 1 bits when it is really a string of all 0 bits. In terms of average member fitness, this function can be described as:

   - f(*...*0*...*) < f(*...*1*...*)
   - f(*...*00*...*) < f(*...*01*...*), f(*...*10*...*) < f(*...*11*...*)
   - Also, f(*...*000*...*) < f(*...*111*...*)

3. F3 (DECEPT2): The purpose of this function is to determine the performance of a GA on a simpler, one bit deceptive problem.

   A sliding window of length 4 is moved one bit at a time over the twenty bit member. The maximum cost is assigned 51.0. At a given window location, each 1 bit in the window subtracts 0.5 from the maximum value. If all bits in the window are 0, 3.0 is subtracted from the maximum value. The minimum cost of the function is 0.0 and occurs when each bit in the individual is 0. When each bit in the individual is 1, the cost is 17.0. For example, the string 11110000111111111111 would have a cost of 22.0.

   This function also contains low-order deceptive schemata, that indicate the function minimum is a string of all 1 bits, when it is really a string of all 0 bits. In terms of average member fitness, this function can be described as:

- f(*...*0*...*) < f(*...*1*...*)
- f(*...*00*...*) < f(*...*11*...*)

4. F4 (MIRROR): The purpose of this function is to examine the ability of a GA to process high-performing, high-order schemata while maintaining the consistency of low-order schemata. The cost function is designed to be much more sensitive to the high-order schemata than the low-order schemata.

   A maximum cost of 39.0 is assigned. Bit 19 of the member is made contiguous to bit 0 for wrap around. For each bit that is the same as its rightmost neighbor, 0.5 is subtracted from the cost. Also, if bit i ($0 \leq i \leq 9$) and bit $i + 10$ differ, 3.0 is subtracted from the cost (e.g. if bit 1 is different from bit 11, 3.0 is subtracted, if bit 2 is different from bit 12, 3.0 is subtracted, etc.). The minimum cost of 0.0 occurs when a string of ten 1's is followed by a string of ten 0's. Since wrap around is allowed, the pattern can begin anywhere in the member. A string of 1's or all 0's has a cost of 20.0

   For example, 00001111111111000000 has cost 0.0 since ten 1's are followed by ten 0's. The string 10101010101010101010 has a cost of 9.0.

   This function should prove difficult for the GA to solve. Consider the member 00011100001110001111. It has a cost of 2.0 yet it is Hamming distance 6 away from the optimal solution. This forms a local optimum from which it is very difficult to escape.

   For this function, the major reduction in cost occurs when high-order schemata are consistent, with a slight reduction when low-order schemata are consistent. As shown in the example, this can lead to members that have strong high-order consistency, but poor low-order consistency. This creates local minima that prove difficult for a GA to avoid or escape.

5. F5 (EIGHTAWAY): This function tests a GA's ability to assemble schemata that are of different order. The function is more sensitive to higher-order schemata than it is to low-order schemata.

   A maximum cost of 41.0 is assigned. For each bit i in the member different from bit $i + 1$, subtract 0.5 from the cost. Also, the cost is reduced as follows:

   (a) For ($0 \leq i \leq 4$)

      i. Let $m = i$, let $n = m + 8$

      ii. Repeat

A. If bit m is different than bit n subtract 2.0 from the cost.

B. Let $m = n$. Let $n = m + 8$

C. If $n \geq 20$, $n = n - 20$

iii. Until $n = i$

The minimum of this function occurs at 01011010101001010101 or its complement. A member of all 0's or all 1's has the maximum value of 41.0

This function forces schema consistency for defining lengths 4, 8, 12 and 16. The function is very sensitive to these high-order building blocks. Consistency should also be maintained for low-order schemata, but must be violated in some instances in order to achieve the optimal string. The local optima this function possesses are similar in nature to those possessed by the function F4 (MIRROR).

6. F6 (ONEMAX): This is the same bit counting function described by Ackley [7] and is unimodal. A maximum cost is assigned 20.0. Each 1 bit subtracts 1.0 from the cost. The minimum cost occurs when all 20 bits are 1 and has a cost of 0.0.

VI. Description of Experiments

The focus of the experiments is to determine the effect of immigration in a Genetic Algorithm on the test suite of functions. As stated earlier, the immigration operator should increase the exploration of a GA without significantly decreasing the exploitation of the GA.

For the steady state GA, the increased exploration should prevent the GA from prematurely converging and becoming trapped in a local minima. This fact should be reflected in the number of GA trials that require an excessively long time to find the function optimum. Also, since the population size is maintained, the GA should have the exploration power of a larger population with the exploitation of a smaller one. This would be reflected in the average number of function evaluations required to find the global optimum. Therefore, for the steady state GA, two performance criteria are examined:

1. The average number of evaluations required to find the function optimum
2. The number of trials that required an excessive number of evaluations to find the function optimum. These trials are called "outliers".

For the restarted GA, the inherent reinitialization process should prevent the population from becoming trapped in a local optimum for many generations. Becoming trapped in a

local optima leads to an excessive number of evaluations, or an outlier. Since the restarted GA should prevent this, immigration should not significantly reduce the number of outliers for an experiment. However, immigration does increase the amount of exploration performed by the GA while the population is converging. The increased exploration may allow the GA to be more circumspect and avoid local minima. Avoiding local minima should decrease function evaluations. Therefore, if immigration aids this algorithm, it would be reflected in the average number of function evaluations required to find the optimum.

A. Design of the steady state GA experiment

Since exploration vs. exploitation is the focus of this work, each function is evaluated over a range of population sizes. The smallest population size is 30. For each function, the population sizes are repeatedly incremented by 10 members until the GA performs worse than with the previous population size.

The fitness function first ranks each member of the size n population. Then, a fitness value is assigned to each member s using the equation:

$$Fitness(s) = exp(3\frac{n - rank(s)}{n})$$

The above fitness assignment curve maps the best member to fitness value 20.0 and the worst member to fitness value 1.0. Using these exponential constants, the best 50 percent of the population has a ratio of 4.5:1 in fitness values. An exponential curve is used to accentuate better performing members while assigning similar fitnesses to poor performers. The curve also prevents high-performing individuals from taking over the population entirely, so the fitness function is not unduly sensitive to cost values. Davis [8] has also used ranked exponential fitness assignment.

The immigration rate m (number of random individuals immigrated each generation) in the experiments ranges from 0 to 4 individuals per generation. Also, a one-point crossover scheme is used.

The probability of child mutation is fixed at 0.005 mutations/bit. Early experimentation determined that the performance of the steady state GA improved when members of the population other than the current children were allowed to mutate. This seemed most important as the population began converging, so a dynamic mutation rate as a function of convergence is used. The dynamic population mutation rate is given by the equation:

$$Mutations/Bit = 0.015(C - 0.5)$$

(where C is the bitwise convergence percentage of the population and ranged from 0.5 to 1.0).

Again, each population member is 20 bits long. Each experiment is assigned a population size and immigration rate, and is tested with 500 separate GA trials, each with a unique random population. Each GA trial counts the number of function evaluations until the global optimum is found. The maximum number of evaluations allowed per trial is 50000. This is performed on all functions in the test suite.

B. Design of the restarted GA experiment.

For this experiment, the population sizes begin at $n = 14$. For each function, the population size is increased by 2 members until the GA performs worse than with the previous population size.

The fitness function first ranks each member of the size n population. Then, a fitness value is assigned to each member s using the equation:

$$Fitness(s) = exp(1.5\frac{n - rank(s)}{n})$$

which assigns fitness values between 1.0 and 4.5. A smaller exponential constant (1.5 instead of 3.0) is chosen for this small population GA. This provides most of the population members with some chance to compete. However, this fitness scheme does enforce a 4.5:1 fitness ratio between the best and worst members of the population.

The immigration rate m in the experiments ranges from 0 to 4 individuals per generation. Also, a one-point crossover scheme is used. The probability of mutation is set at 0.005 mutations/bit.

For functions F1 - F5, the threshold convergence ratio is set 0.85. In other words, when the population is 85 percent bitwise converged, the two best members are kept and random members fill the remainder of the population. This convergence ratio was selected after some experimentation with values of 0.75 and 0.95. In general, for functions F1 - F5, a convergence value of 0.75 forced the GA to import random members before good structure had been developed, and led to an increased number of function evaluations. A convergence value of 0.95 often required convergence of members to a local optimum which was already present in the population. This also led to an excessive number of function evaluations.

For F6, a unimodal function, a convergence ratio of 0.95 provides the best performance. For this function, the population could not settle into a local optimum and could continue useful schema processing to a higher degree of convergence.

As with the steady state GA, each population member is 20 bits long. The same experimental constraints are also present.

VII. Experimental Results

The experiments provided a measure of difficulty for each of the six functions in the suite. Figures 1a-1f present a comparison of the performance of the steady state GA with immigration (dashed bars) and without immigration (solid bars) on functions F1 - F6 of the test suite. Figures 8a-8f present a comparison of the performance of the restarted GA. These plots reflect the average number of evaluations required by the GA to find the optimum value of each function. The number of immigrations per generation that produced these results is labeled in each figure.

Based on the experimental results, both GA's easily solved function F6 (ONEMAX) which is a unimodal function. This was to be expected. Functions F1 (ODDEVEN) and F3 (DECEPT2) proved only a little more difficult to the GA's. This indicates that the GA does a good job assembling low-order optimal schemata into an optimal string. It also indicates that the GA can overcome some deception in its search.

Function F4 (MIRROR) was next in level of difficulty, followed at a distance by F5 (EIGHTAWAY). Both of these functions required the development of high-order schemata and the consistency of low-order building blocks. The difficulty of the GA in achieving the global optimum of each function may be due to the crossover operation used. The strength of one-point crossover is its ability to assemble low-order building blocks into optimal strings. High-order schemata have a greater chance of being destroyed, as was demonstrated by these experiments. Perhaps the performance of the GA's would improve using a crossover mechanism that is less positionally biased.

Function F2 (DECEPT1), a two-bit deceptive function, proved extremely difficult to both GA's. When compared to function F3, a one-bit deceptive function, one can see that increased deception has a profound effect on the optimization capabilities of a GA.

The next sections describe in detail the experimental results, and examine effects of immigration on a Genetic Algorithm.

A. Steady state GA

The smallest average number of evaluations for functions F1 - F5 occurred when immigration was present. This was expected, since these functions contain many local optima. For

function F6, the steady state GA without immigration outperformed the modified GA. Again, this was predicted, since F6 is a unimodal function.

It is important to note that these figures provide the best results of the steady state GA with immigration. The immigration rate that performed best for a function is called the "optimal" immigration rate for that function. In general, all immigration rates (greater than 0) up to and including the optimal rate resulted in an improvement in performance over the non-modified GA.

For function F1, (Fig. 1a) the GA without immigration produced the best results at a population size of 80 and required an average of 1688 function evaluations to find the optimum solution. With 2 immigrations per generation at a population size of 60, the GA required only 1352 function evaluations. Therefore, the GA without immigration resulted in a 24.8 percent increase in search time over the GA with immigration. In fact for each function F1 - F5, *immigration resulted in a reduction in the number of function evaluations.*

Examining Figure 1, it is interesting to note that for most of these functions, the reduction in function evaluations using immigration is largest for small populations, and decreases as the population size increases. This indicates that the smaller populations require the added exploration that immigration provides, while larger populations possess sufficient exploration power.

Also, in most trials shown in Figure 1, the GA with immigration performed better than a GA with 10 more population members without immigration. This provides more evidence that immigration allows the GA to search the space of a larger population.

Further, in F1, F3 and F5 the optimum with immigration occurs in smaller populations than the optimum without immigration. This lends credence to the theory that immigration allows small populations to retain their selection pressure. This is demonstrated further in Figures 2-7.

The histograms in Figures 2-7 present the results of 500 GA trials on each function with and without immigration. The figures present the number of trials (Y axis) that require a given number of function evaluations (X axis) to find the global optimum for a range of population sizes. The different population sizes are represented by solid, dashed and dotted lines. Figures 2a - 7a show results of the GA without immigration. Figures 2b - 7b show the results with optimal immigration. For example, in Figure 2a, the GA without immigration and a population size of 40 (solid line) found the optimal solution in 200 function evaluations (X axis) in 129 out of its 500 separate trials for function F1.

In order to reduce the length of the X axis, all trials that require an excessive number

of function evaluations are grouped together at the last point on the X axis. For example, in Figure 2a, the GA with no immigrations and a population size of 40 had 78 points that required more than 5000 evaluations to find the optimal solution. These points are referred to as "outliers".

Examining the results of the GA without immigration (2a - 7a), one can see that the peak generally shifts to the right with increasing population size. This demonstrates the decrease in selection pressure, which inhibits the GA from finding easy solutions quickly. From these figures, one also notes that increasing population size reduces the number of outliers. This indicates that an increase in population size increases the exploration power of the GA.

Let us now compare the GA without immigration to the GA with immigration. As shown in Fig. 2, the GA with immigration (Fig. 2b) has significantly fewer outliers than the GA without immigration (Fig. 2a). This is an example of how immigration can increase exploration. Further, the peaks of Fig. 2b occur at about the same number of function evaluations (X axis) as the peaks in Fig. 2a. As discussed above, if selection pressure was decreased by immigration, we could expect the peaks in Figure 2b to be shifted right of the peaks in 2a. This is not the case, so *the GA with immigration maintains selection pressure and exploitation power.*

Figures 3 and 4 present similar results. Further, for these functions the magnitude of the peaks actually increased with immigration. This is due to the reduction in the number of function evaluations for trials to the right of the peaks, another indication of increased exploration.

Figures 5 and 6 show the peaks occurring near the same X axis location, but again show that immigration does not always eradicate all the outliers for various population sizes. It does show, however, that immigration still reduces the number of outlying trials. Since these outliers contribute heavily to the average number of function evaluations required to find the optimum (Figure 1a-f), it is clear that eliminating outliers reduces this value.

Figure 7 shows the result of function F6 with no immigrations and with 1 immigration. These experiments verified our prediction that a steady state GA with immigration would perform poorly on a unimodal function. As shown by the plot, immigration shifted the peak to the right and reduced it. The average number of evaluations rose from 358 (without immigration) to 461 (with immigration). In this case, immigration did not achieve the balance between added evaluations and missing population structure. This indicates that immigration is not necessary for functions in which the structure can be selected reliably and propagated easily through the population of a steady state GA.

Overall, these experiments show that a population size between 60 and 70 members performs best for the steady state GA described above. To improve performance, an immigration rate of 2 or 3 members per generation should be used on functions that contain difficult local optima.

B. Restarted GA

As with the steady state GA, the smallest average number of evaluations for functions F1 - F5 occurred when immigration was present in the restarted GA. This was predicted, since these functions contain many local optima.

However, for function F6, the unimodal function, the restarted GA with immigration outperformed the GA without immigration. This indicates that adding random members during convergence improves the efficiency of a restarted GA over both unimodal and mutimodal functions. This phenomena is quite interesting, and should be studied in further detail.

Again, these figures provide the best results of the restarted GA with immigration. In general, all immigration rates (greater than 0) up to and including the optimal rate resulted in an improvement in performance over the non-modified GA.

These experiments show that a population size between 16 and 20 members performs best for the restarted GA described above. To improve performance, an immigration rate of 2 or 3 members per generation should be used on functions that contain difficult local optima. Unimodal functions are more efficient at smaller populations, and can also be benefit from the effects of immigration.

VIII. Conclusions

This study has examined the tradeoff between exploration and exploitation in Genetic Algorithms. It proposed that a GA could increase exploration power while maintaining selection pressure by replacing poor performing individuals in a population with random members.

The results of the experimentation show that immigration improves the performance of steady state GA's when optimizing functions that contain local optima which are difficult to avoid or escape. The experiments also show that immigration improves performance for a spectrum of functions using a restarted GA.

Although these experiments are complete, there are several recommendations for future work. Testing immigration on a generational GA would be the next logical step of this

research. Also, a further examination of immigration vs. restart would be in order. Finally, testing functions F4 (MIRROR) and F5 (EIGHTAWAY) using a less positionally biased crossover operator, such as uniform crossover, would prove interesting.

In conclusion, this study demonstrates that immigration is a viable operator for improving the efficiency of GA's on difficult optimization problems.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[2] D. E. Goldberg, "Sizing populations for serial and parallel genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 70–79, 1989.

[3] K. A. De Jong, *An Analysis of the Behavior of a class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

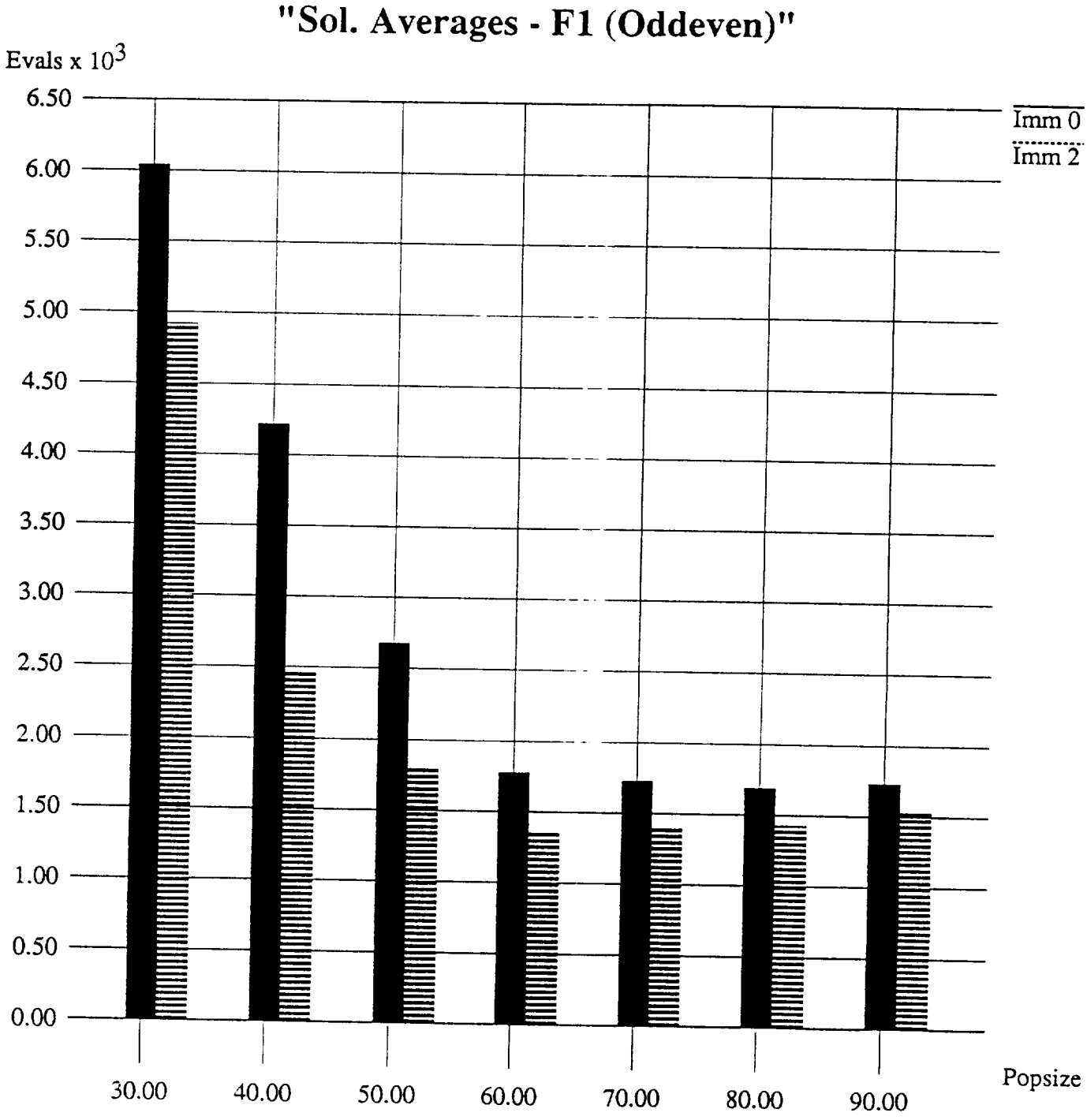
[4] J. D. Schaffer, R. A. Caruna, L. J. Eschelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 51–60, 1989.

[5] M. C. Moed and G. N. Saridis, "A Boltzmann machine for the organization of intelligent machines," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, September 1990.

[6] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, (Cambridge, MA), pp. 14–21, 1987.

[7] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, 1987.

[8] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 61–69, 1989.



2 IMM/GEN
Fig 1a

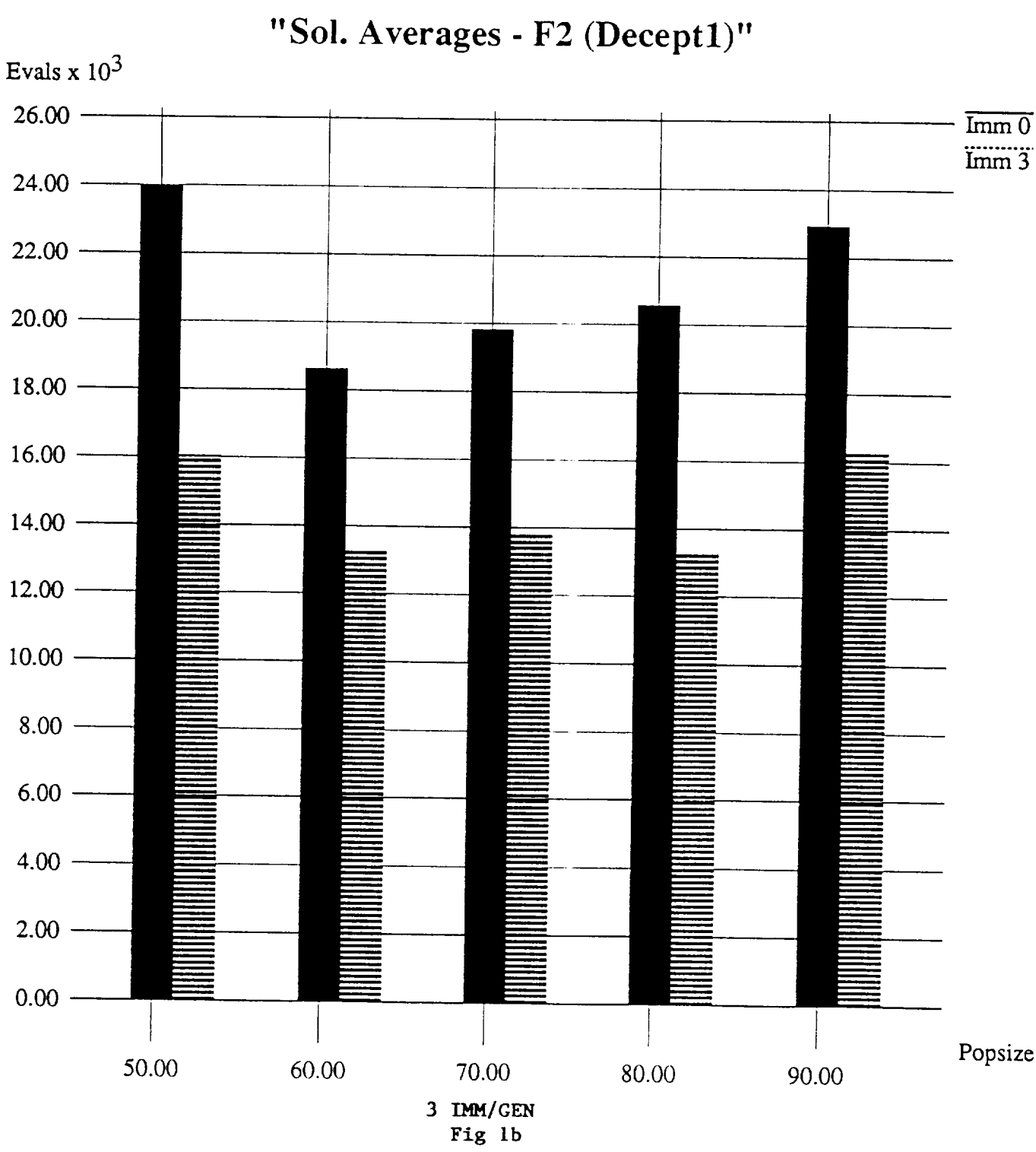
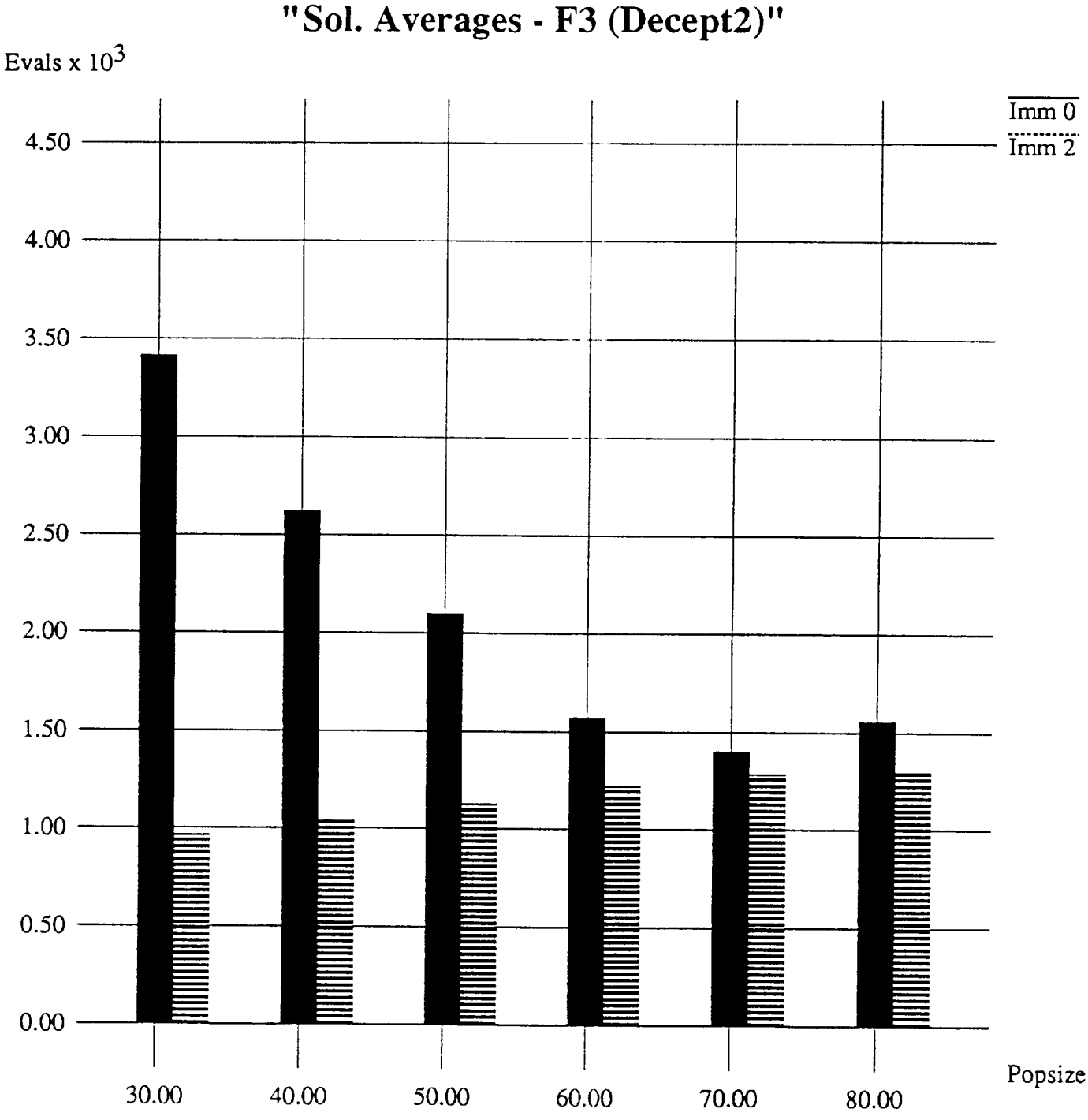

"Sol. Averages - F2 (Decept1)"
Evals x 10^3
26.00
24.00
22.00
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Imm 0
Imm 3
50.00
60.00
70.00
80.00
90.00
Popsize
3 IMM/GEN

Fig 1b



2 IMM/GEN
Fig 1c

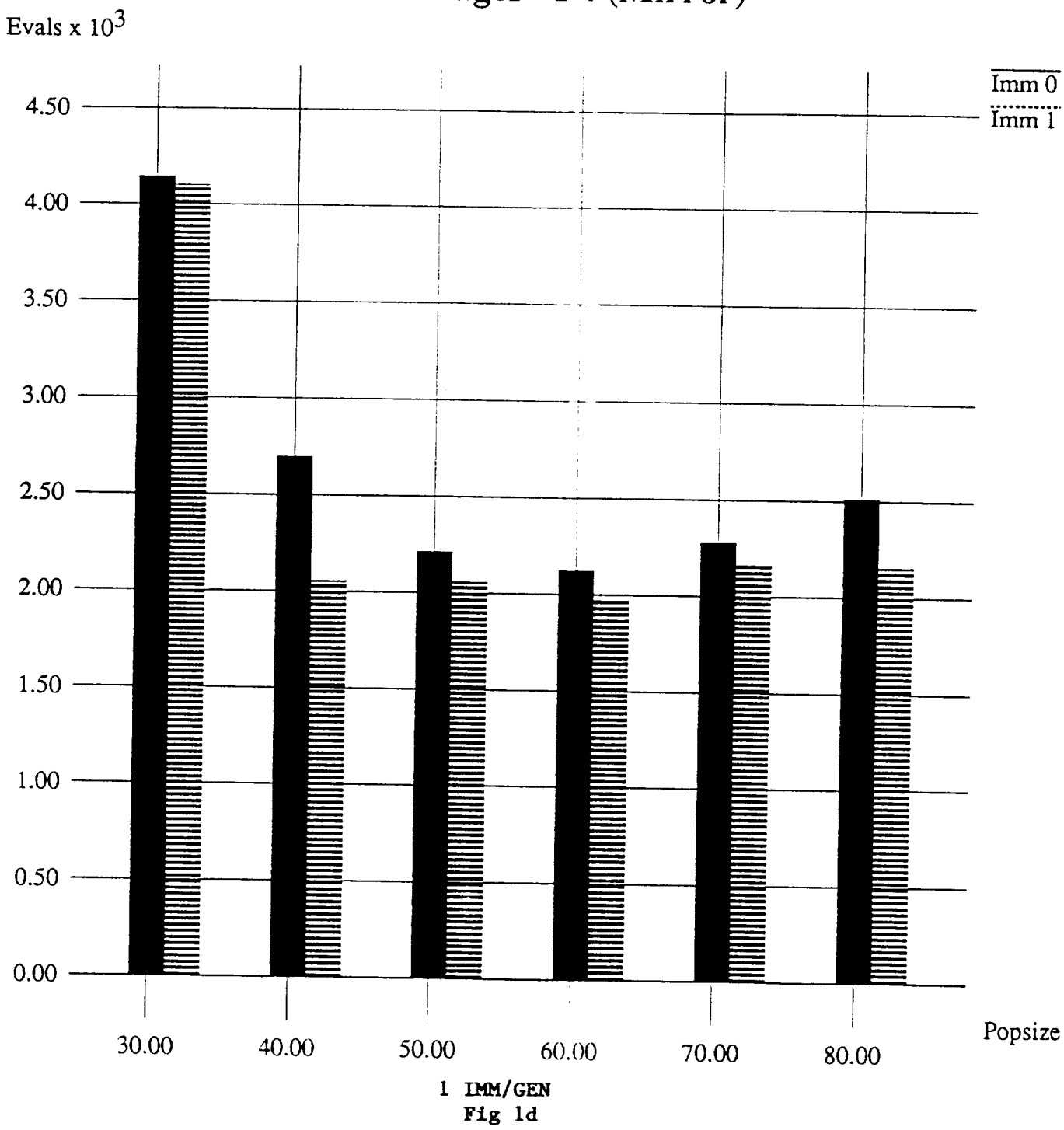
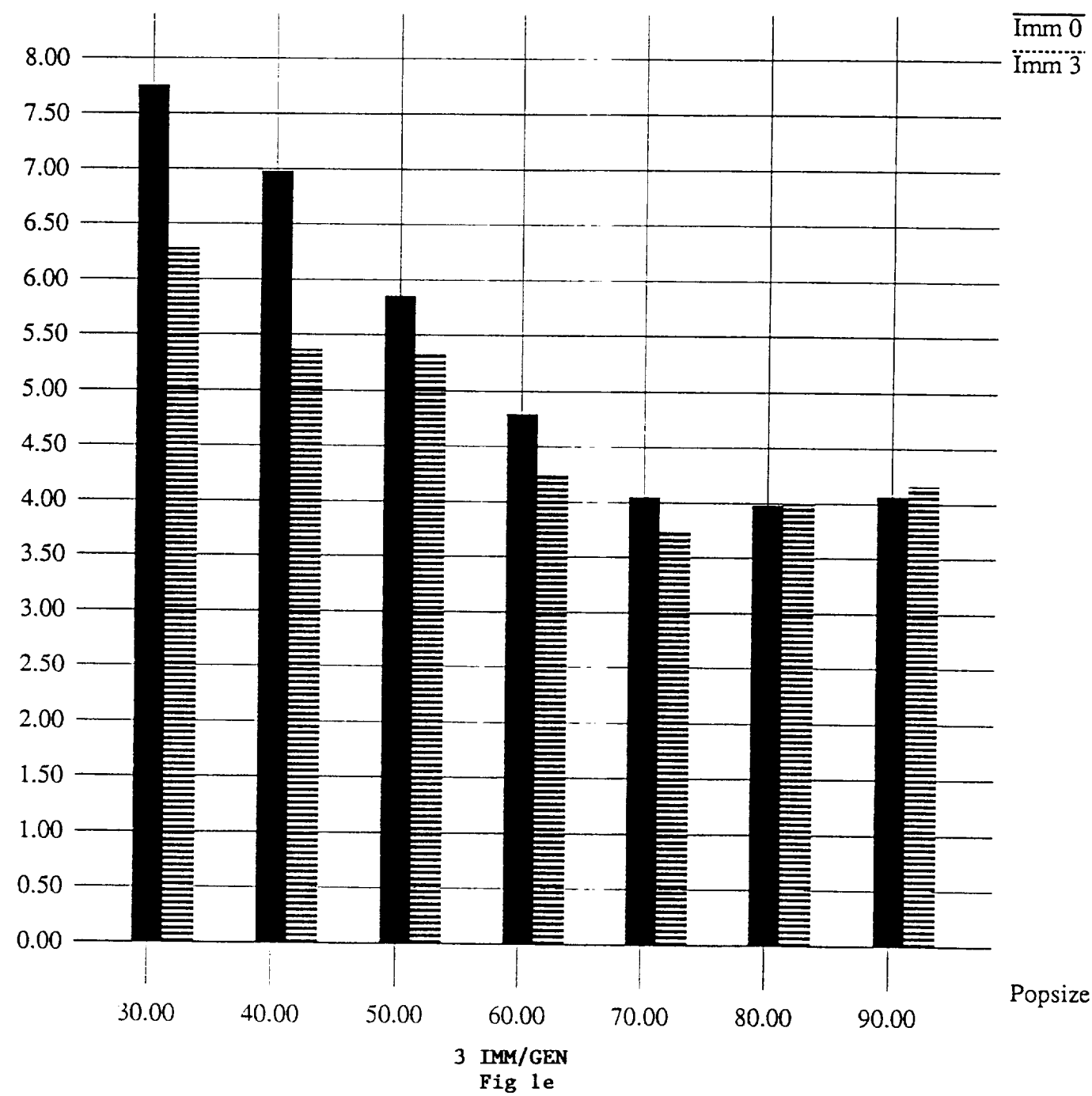
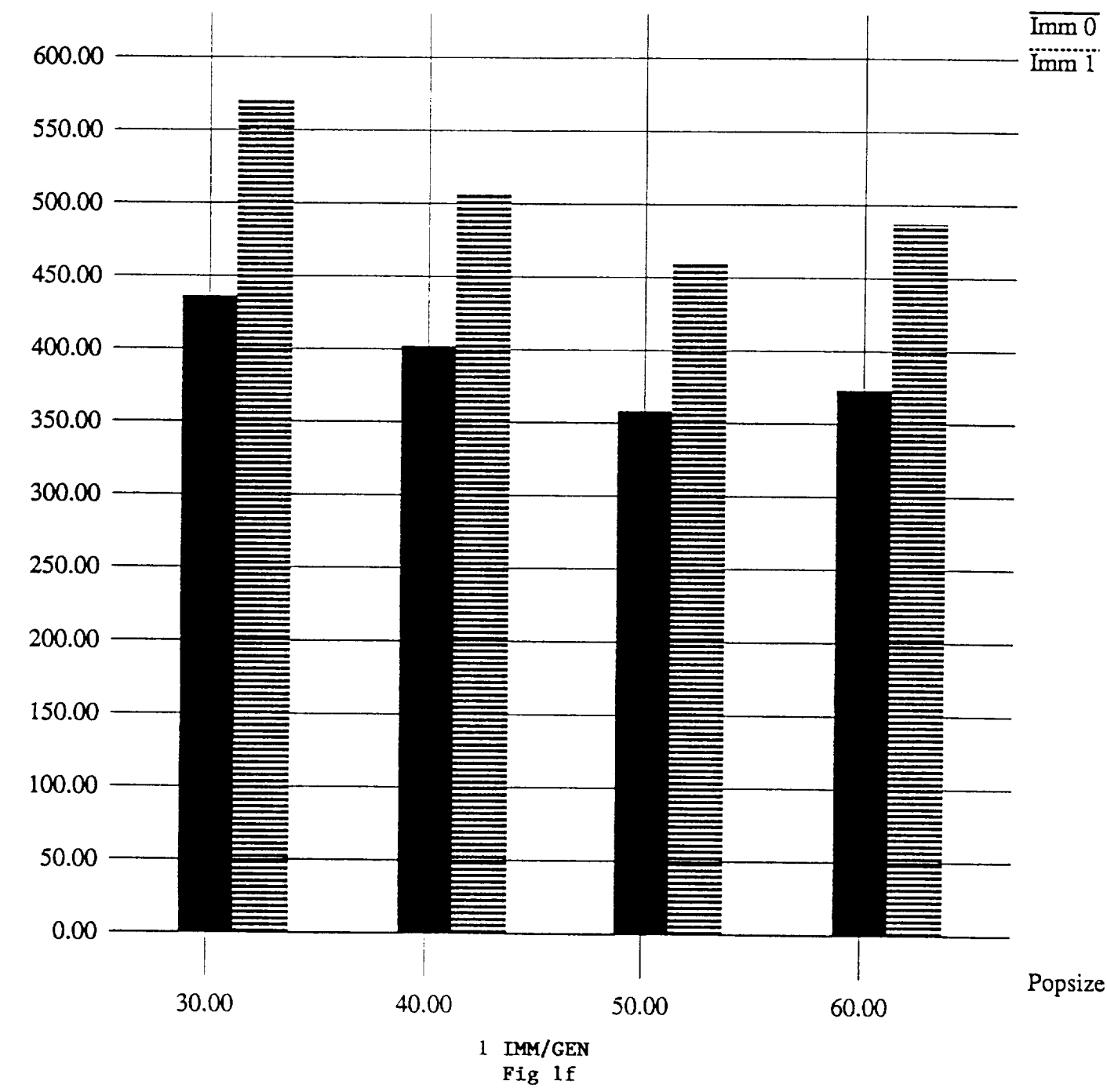

"Sol. Averages - F4 (Mirror)"
Evals x 10^3
Imm 0
Imm 1
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
30.00
40.00
50.00
60.00
70.00
80.00
Popsize
1 IMM/GEN

Fig 1d

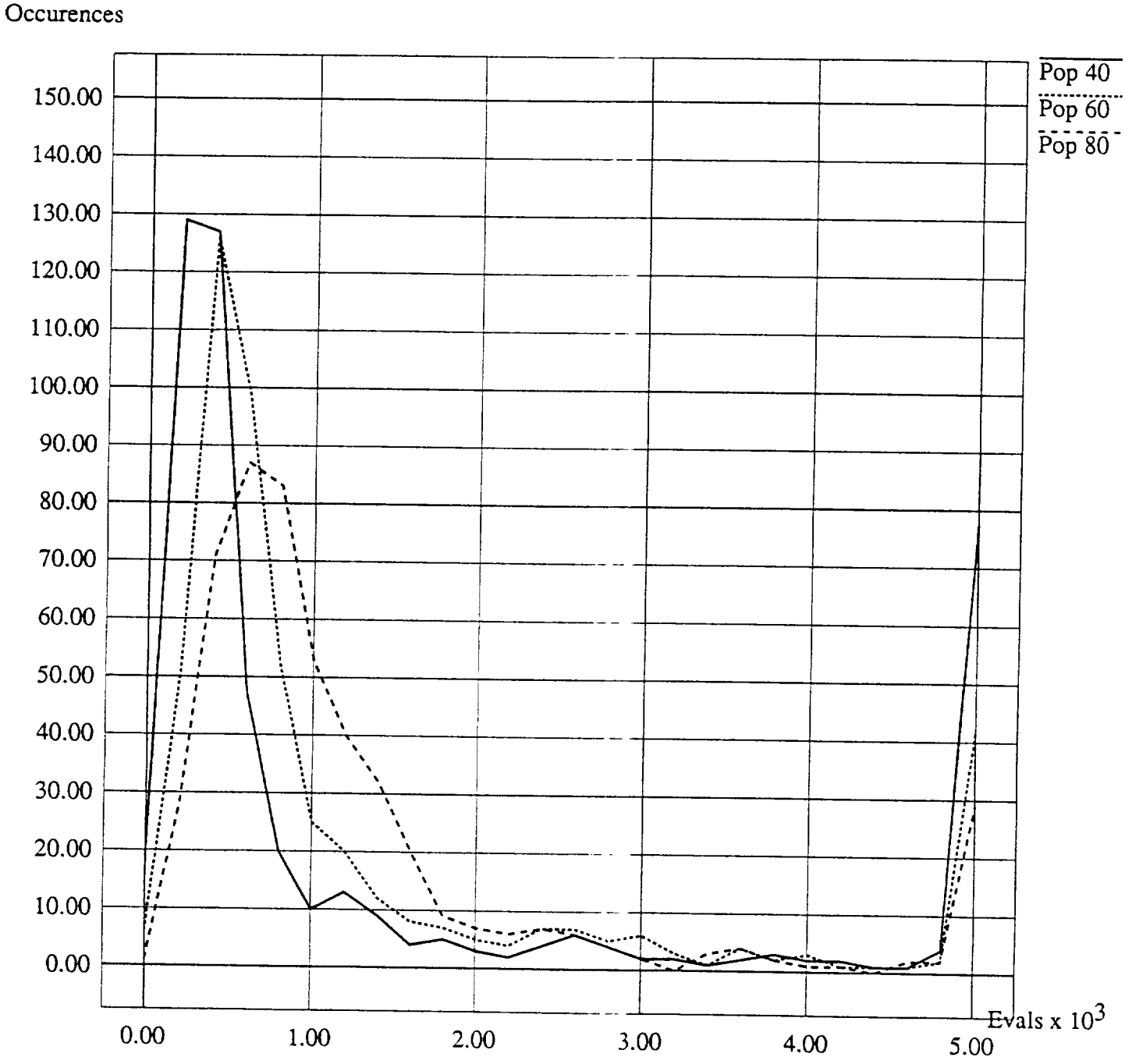


3 IMM/GEN
Fig 1e



1 IMM/GEN
Fig 1f

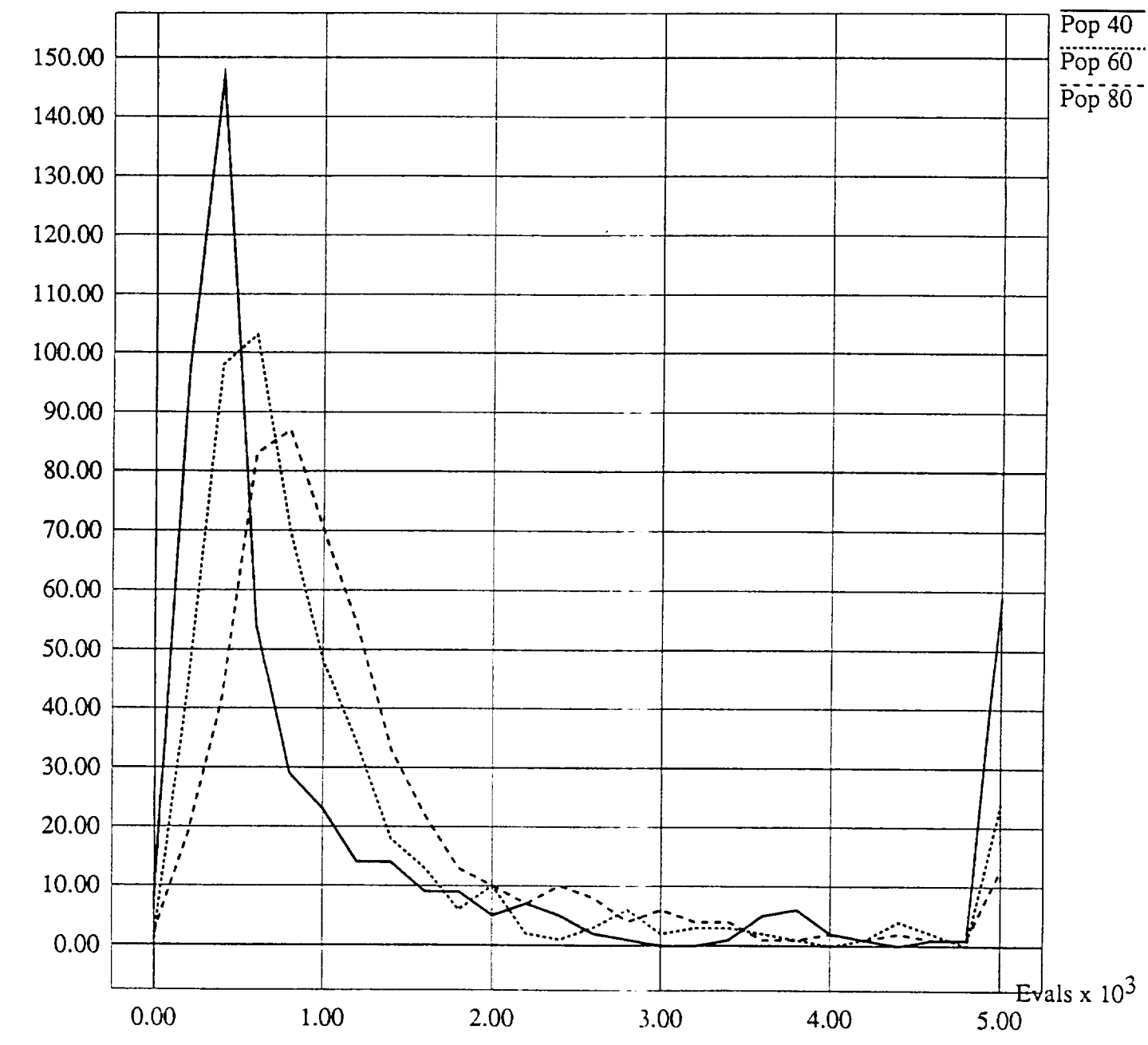
F1 (ODDEVEN)



0 IMM/GEN

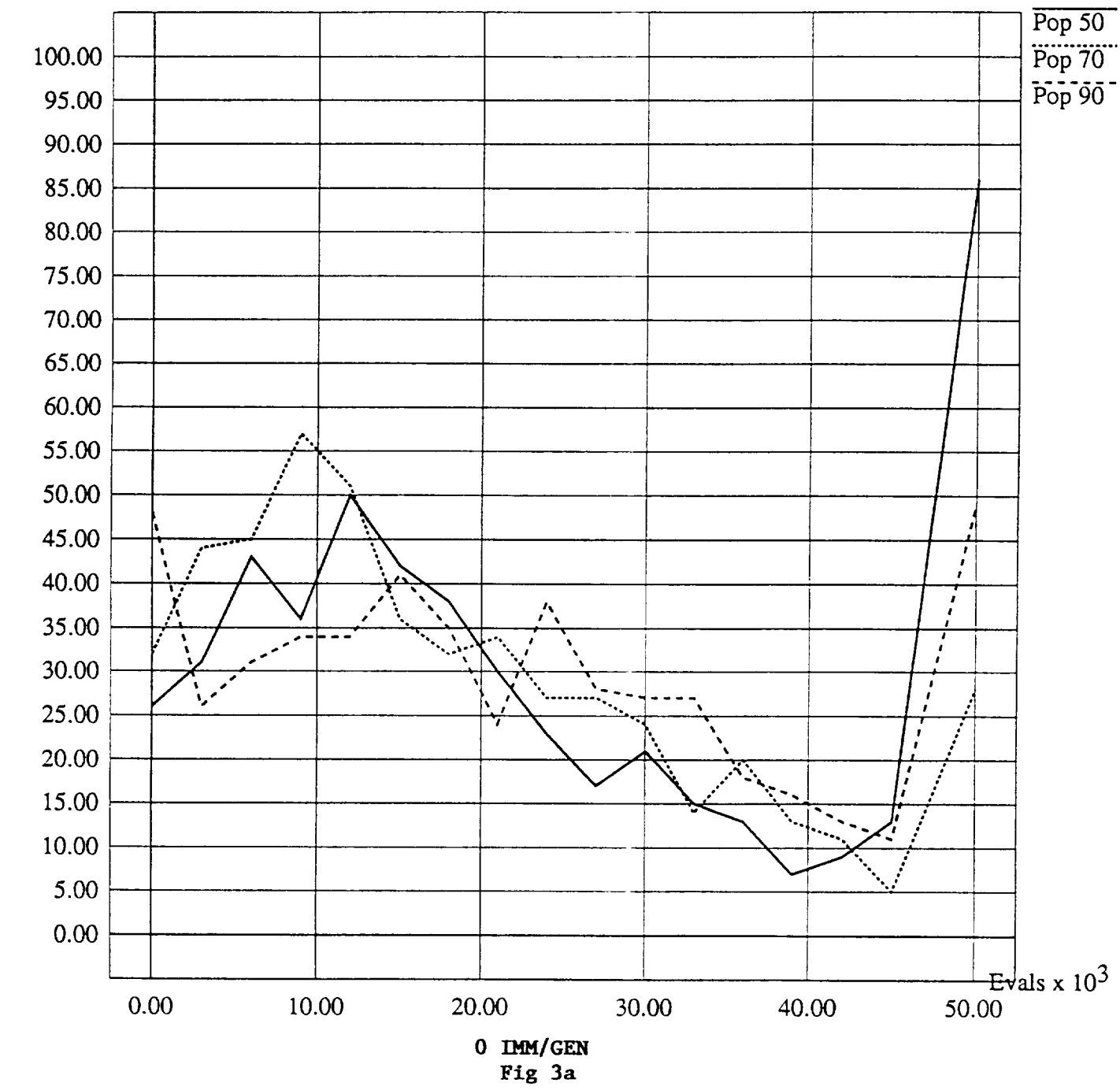
Fig 2a

F1 (ODDEVEN)



2 IMM/GEN
Fig 2b

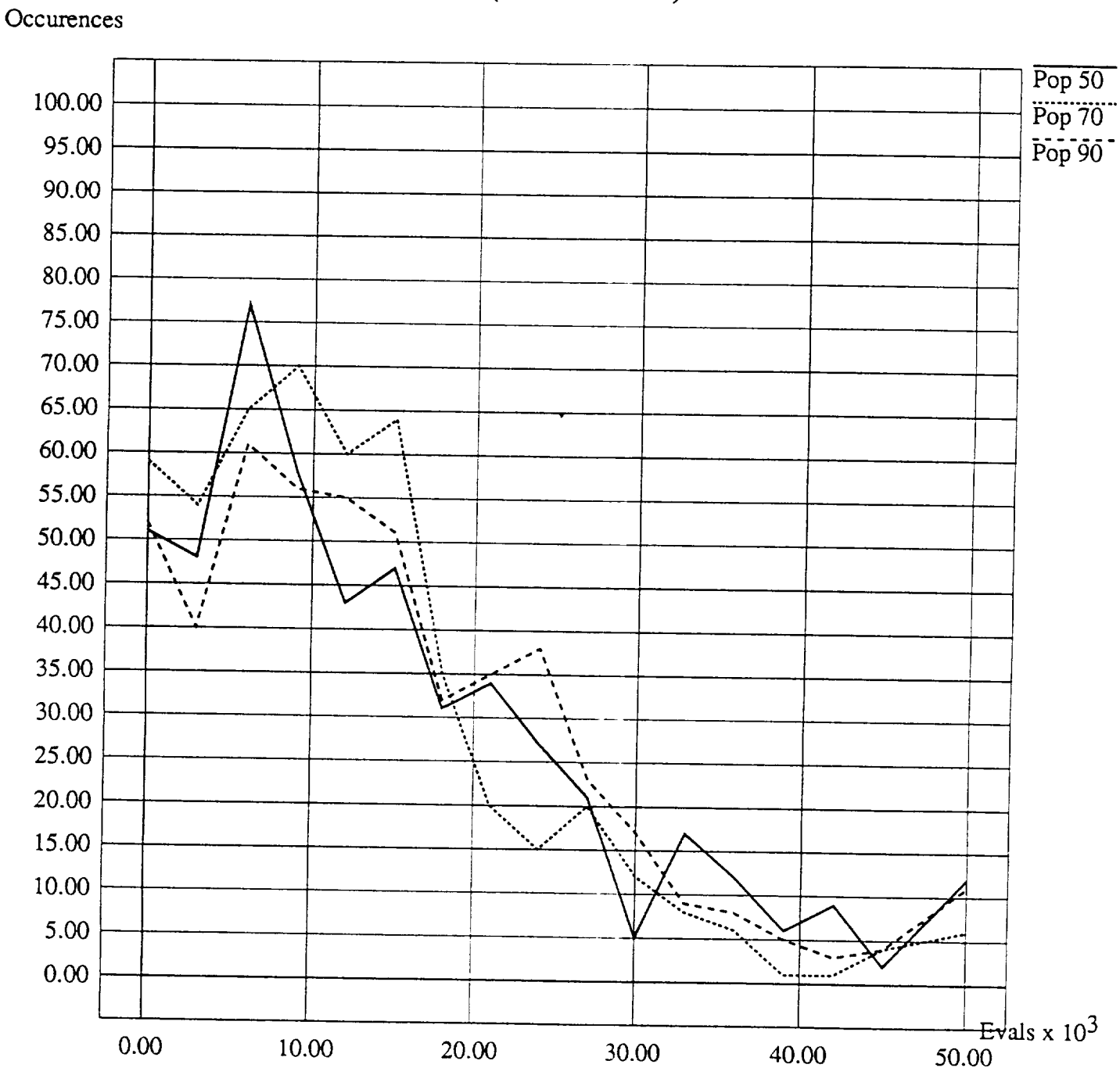
F2 (DECEPT1)



0 IMM/GEN

Fig 3a

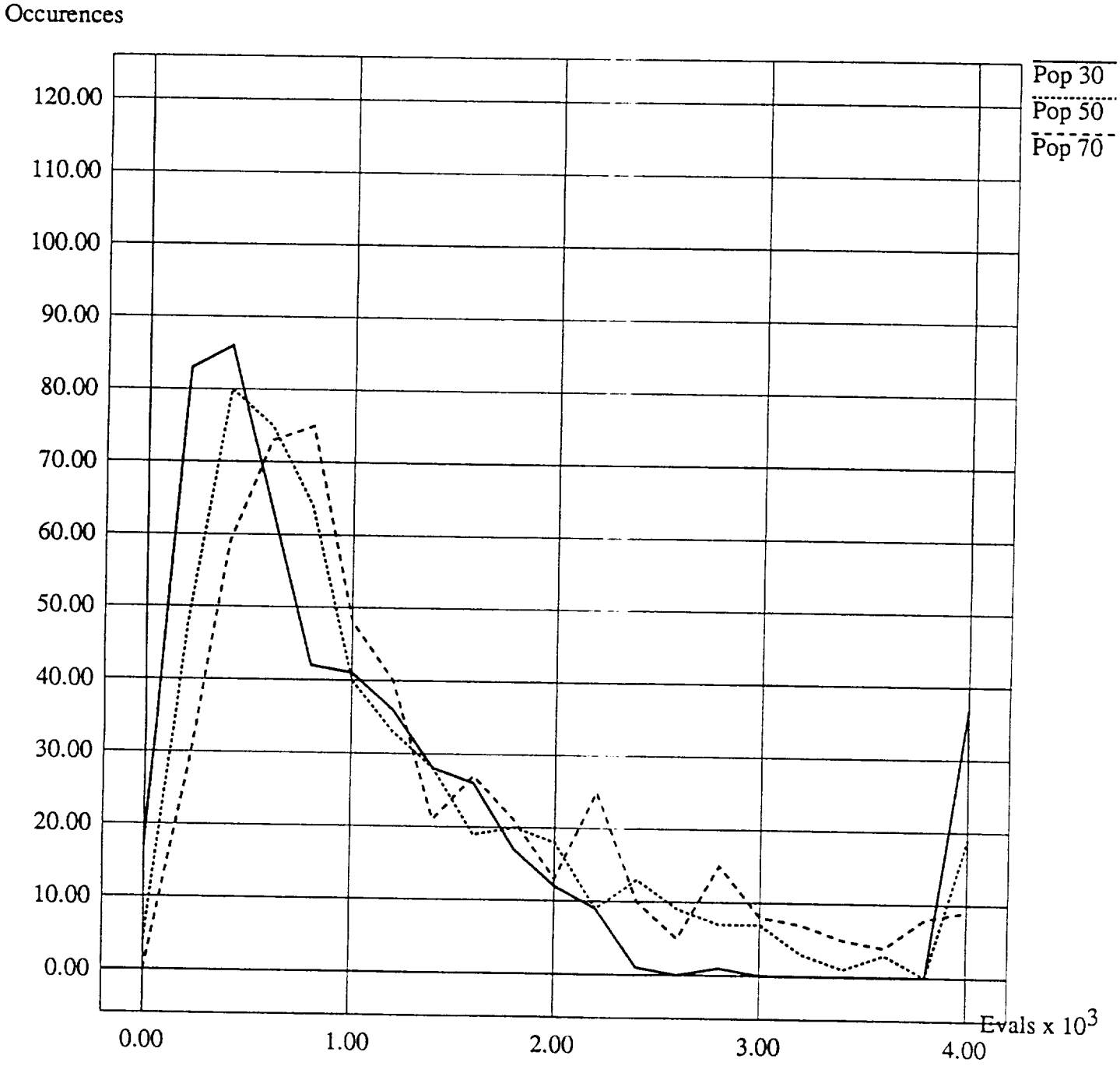
F2 (DECEPT1)



3 IMM/GEN

Fig 3b

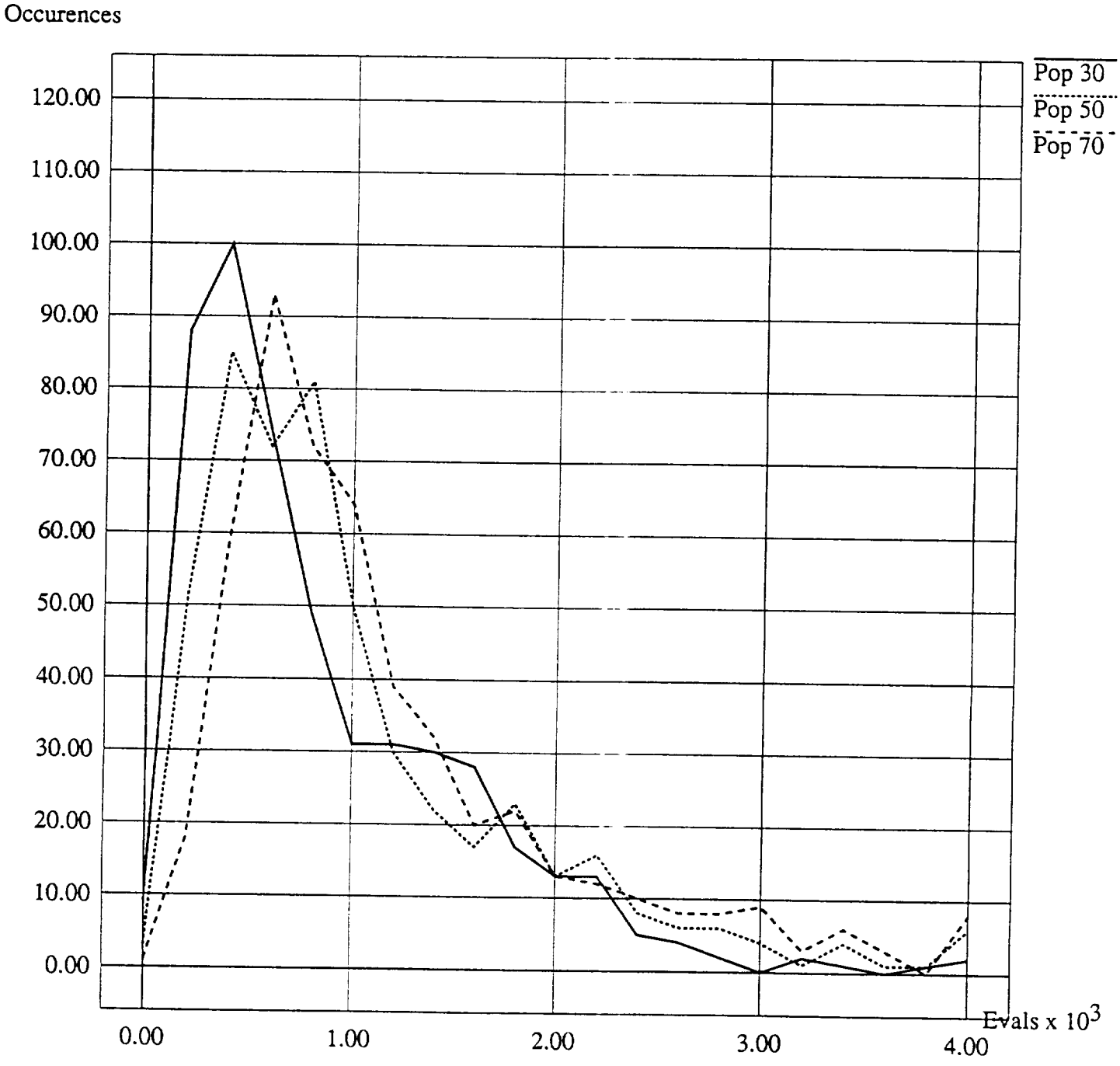
F3 (DECEPT2)



0 IMM/GEN

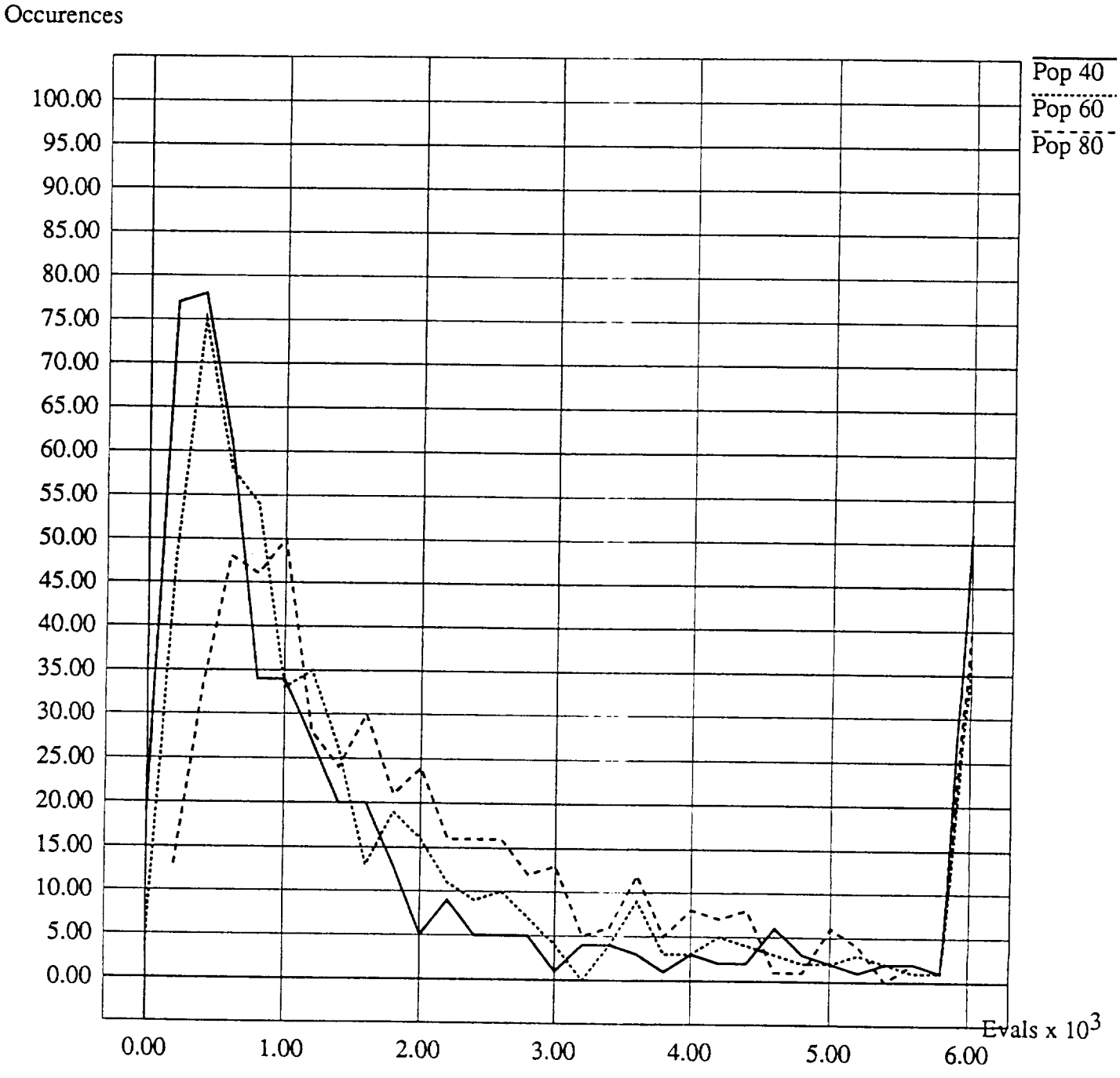
Fig 4a

F3 (DECEPT2)



2 IMM/GEN
Fig 4b

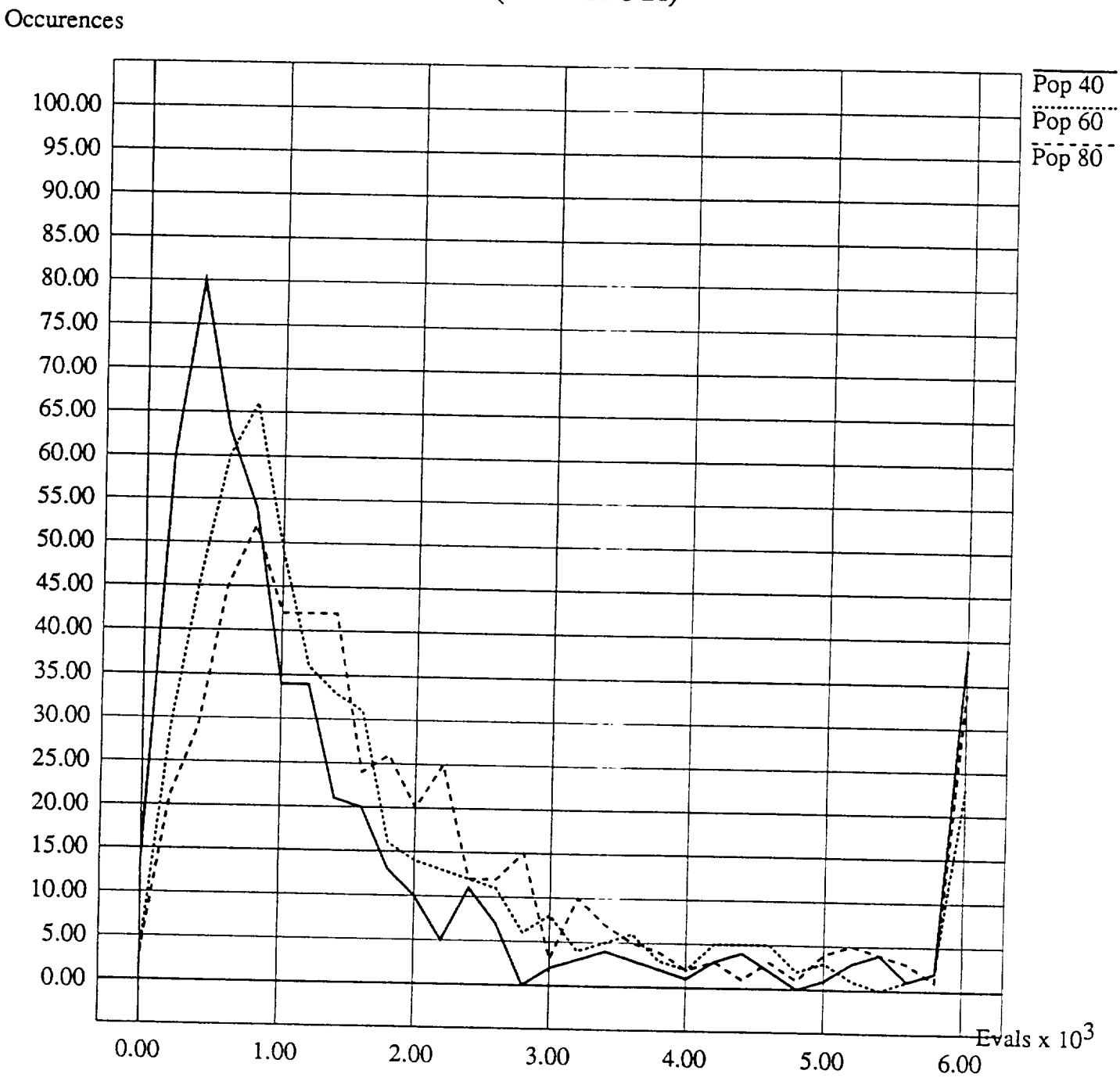
F4 (MIRROR)



0 IMM/GEN

Fig 5a

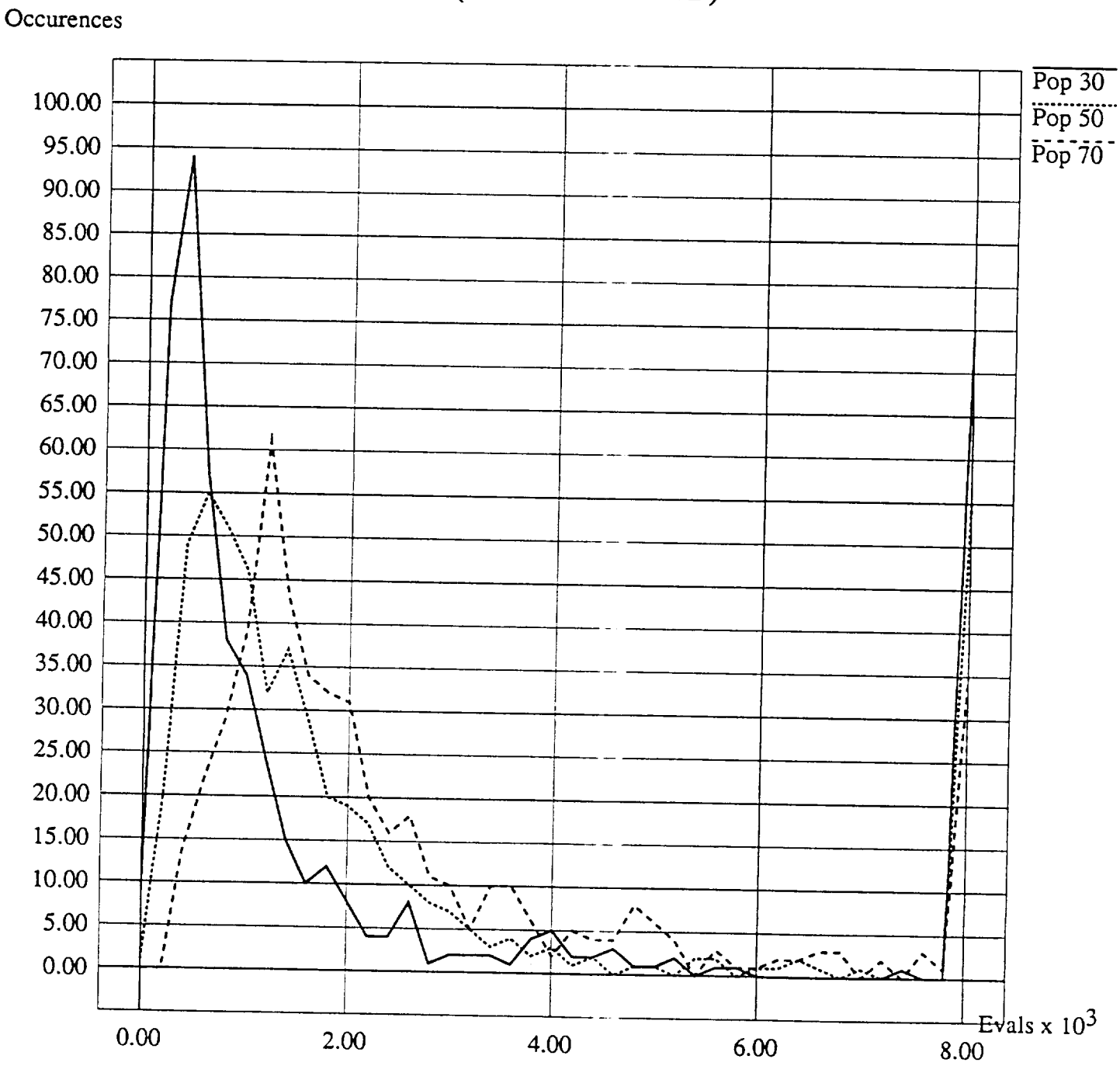
F4 (MIRROR)



1 IMM/GEN

Fig 5b

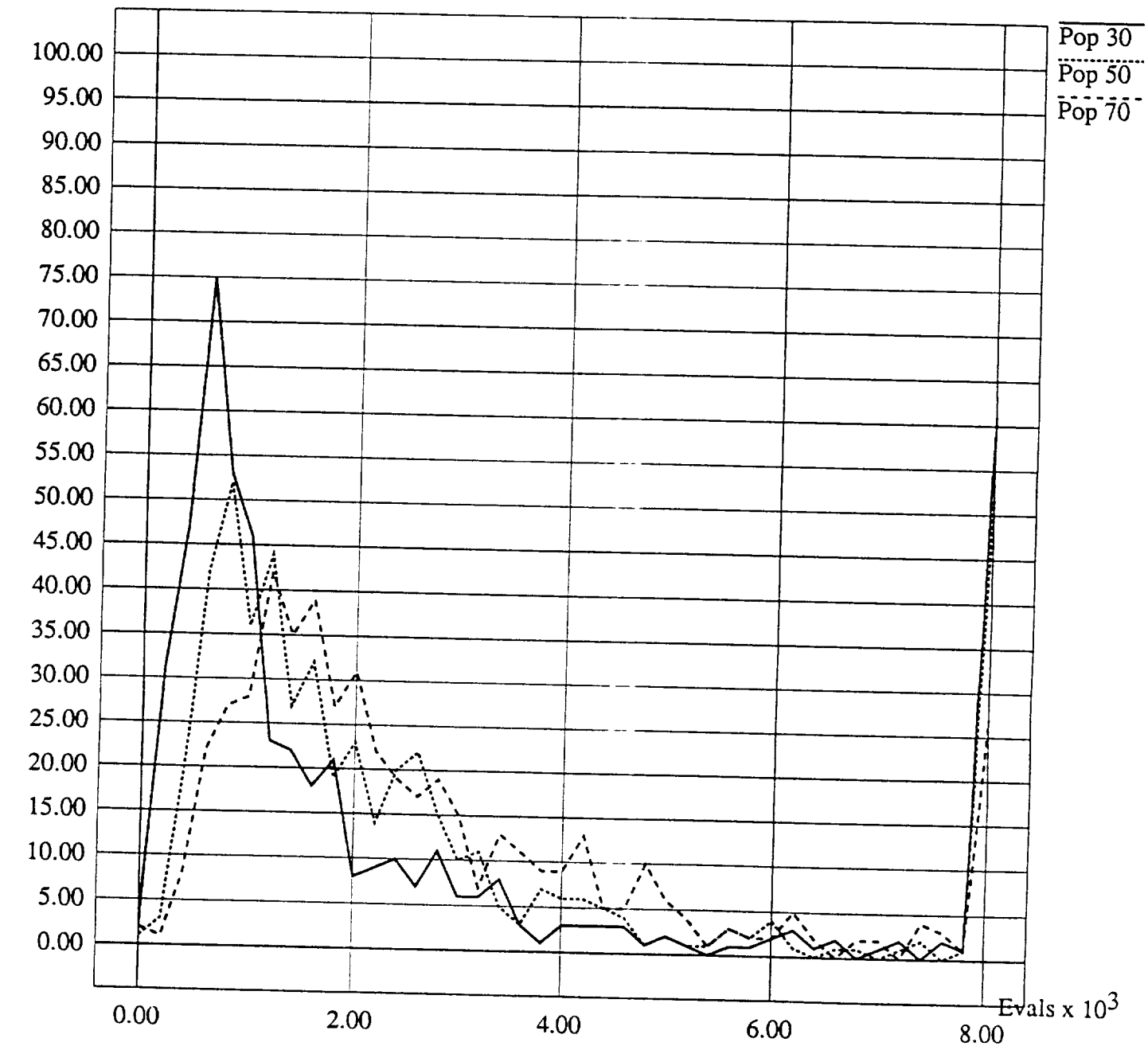
F5 (EIGHTAWAY)



0 IMM/GEN

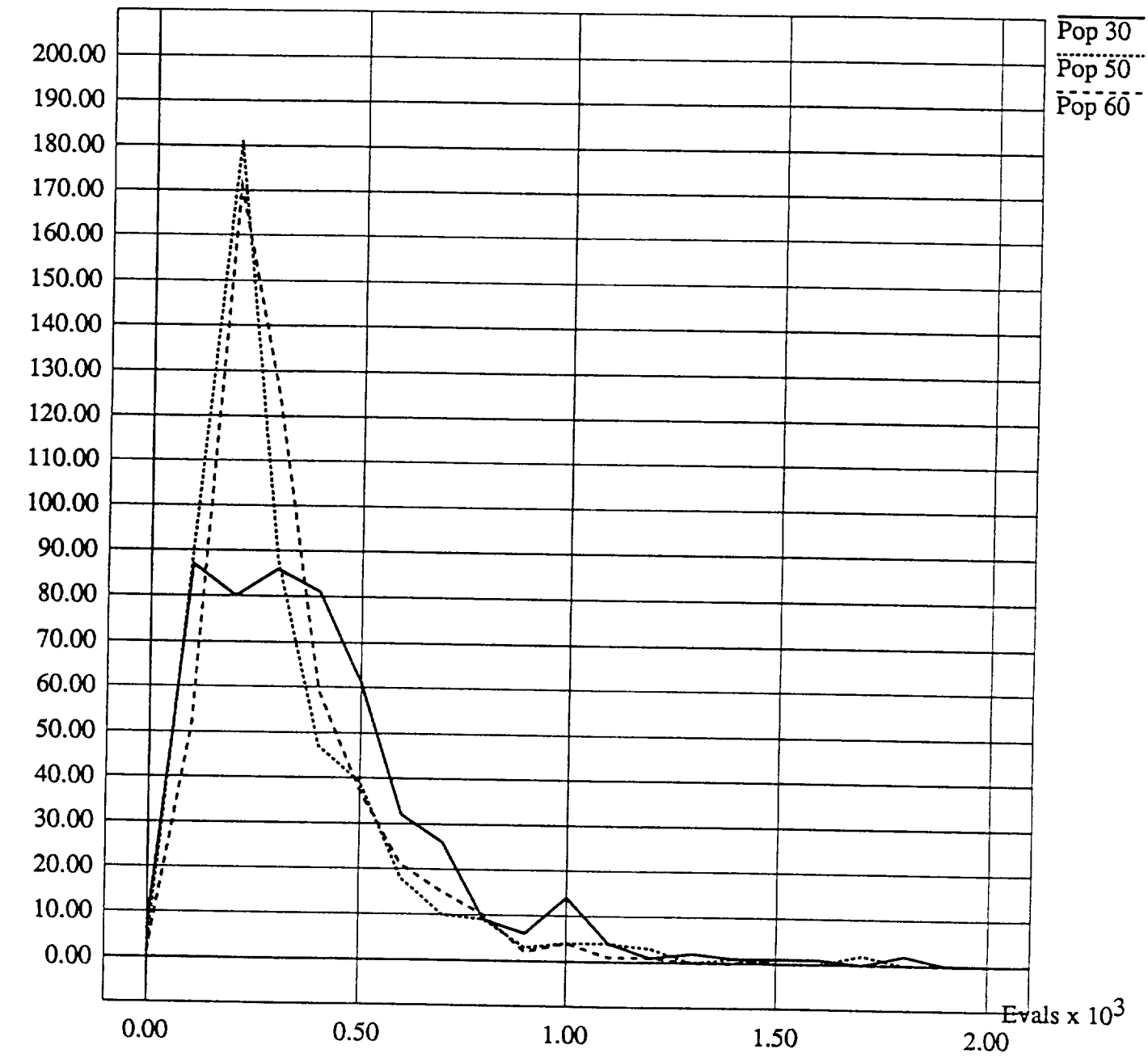
Fig 6a

F5 (EIGHTAWAY)



3 IMM/GEN
Fig 6b

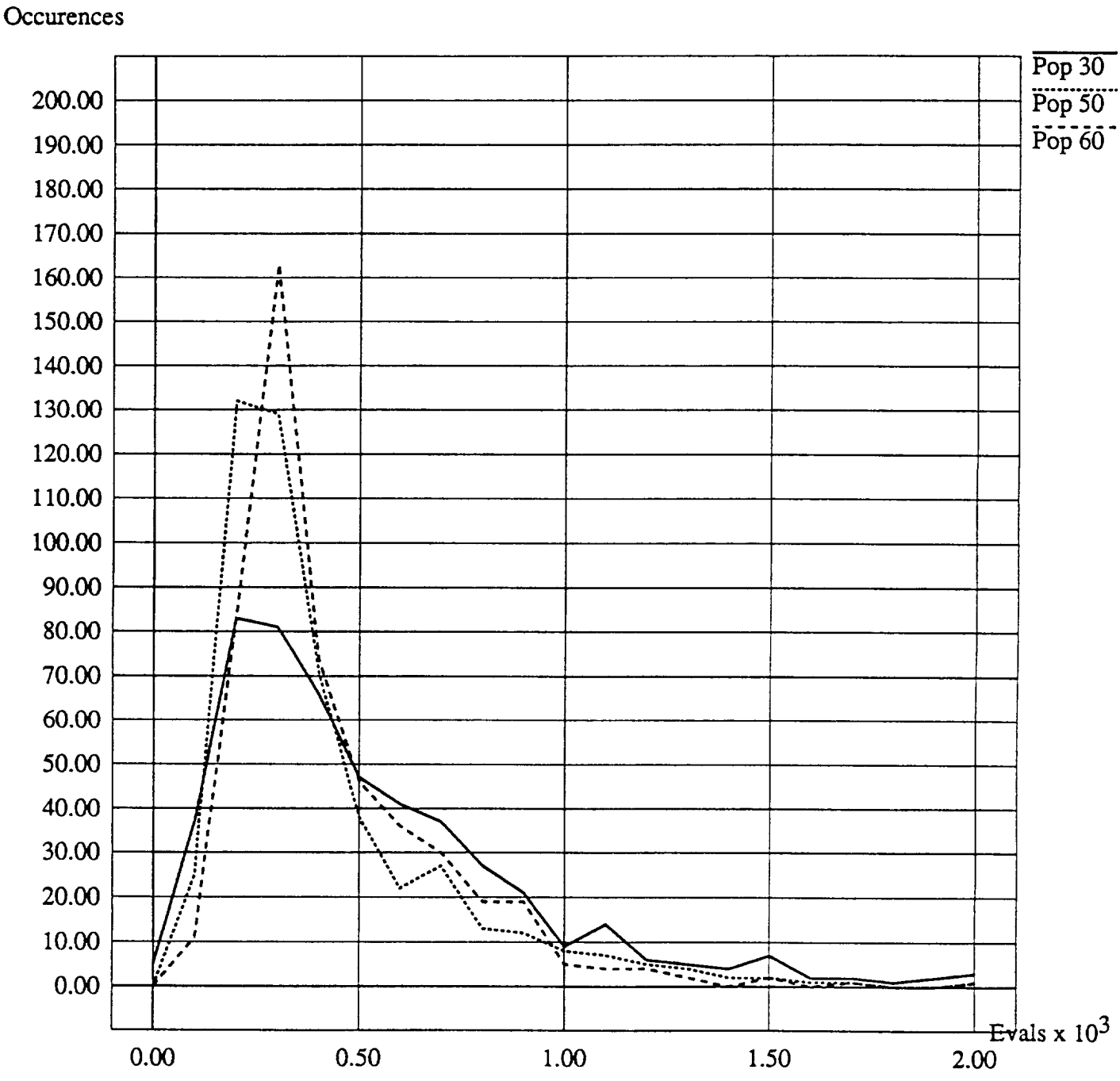
F6 (ONEMAX)



0 IMM/GEN

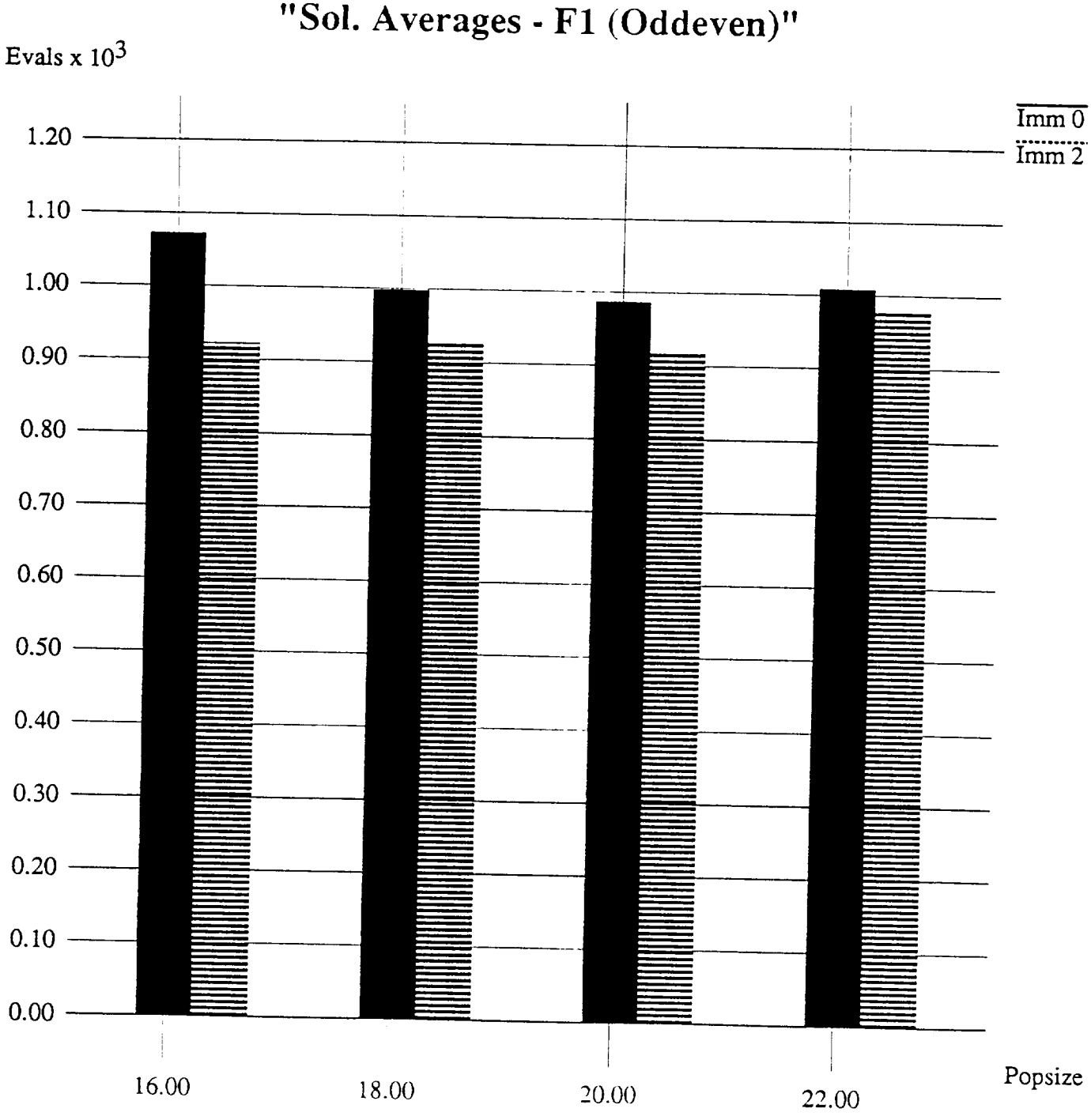
Fig 7a

F6 (ONEMAX)

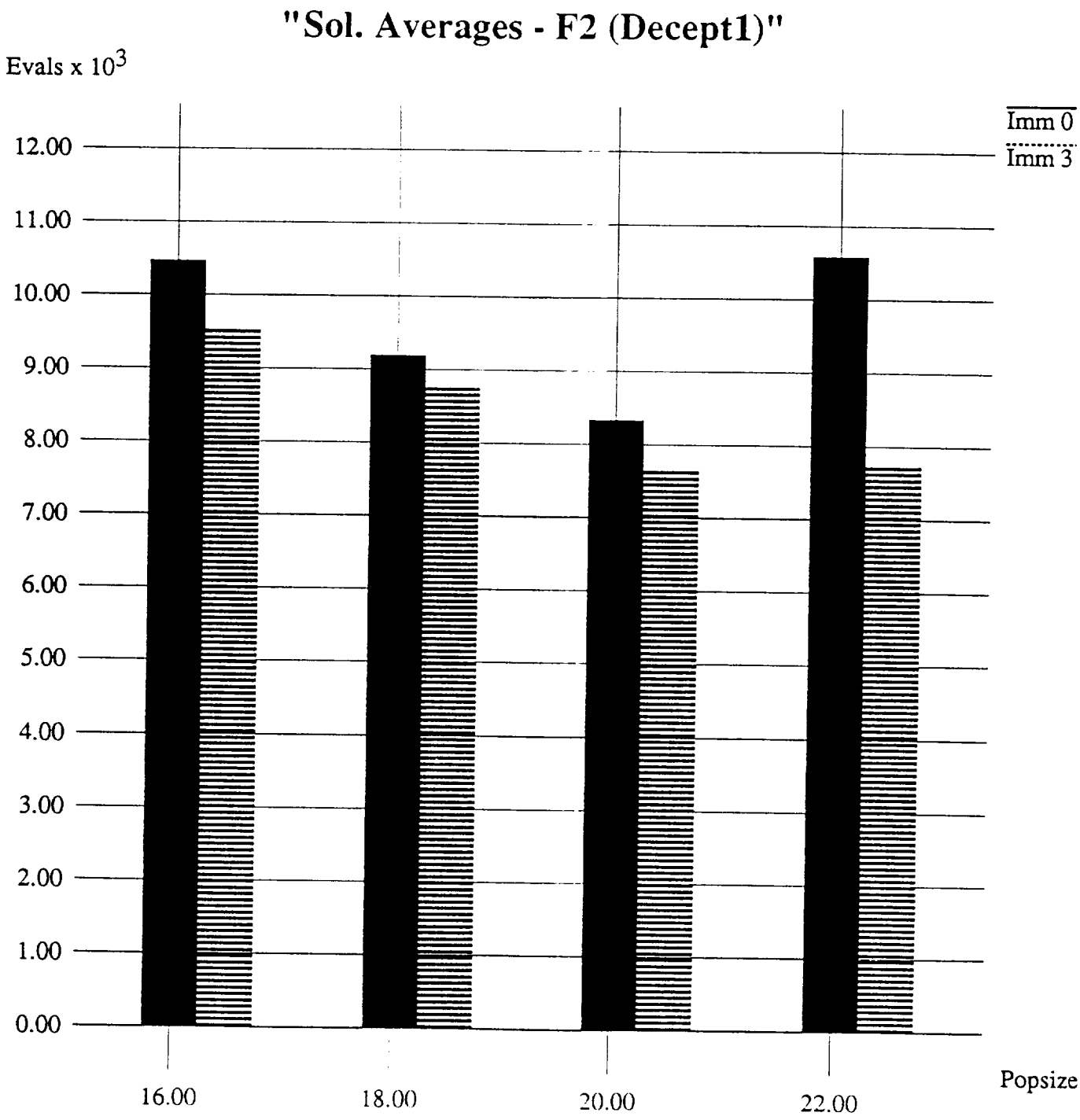


1 IMM/GEN

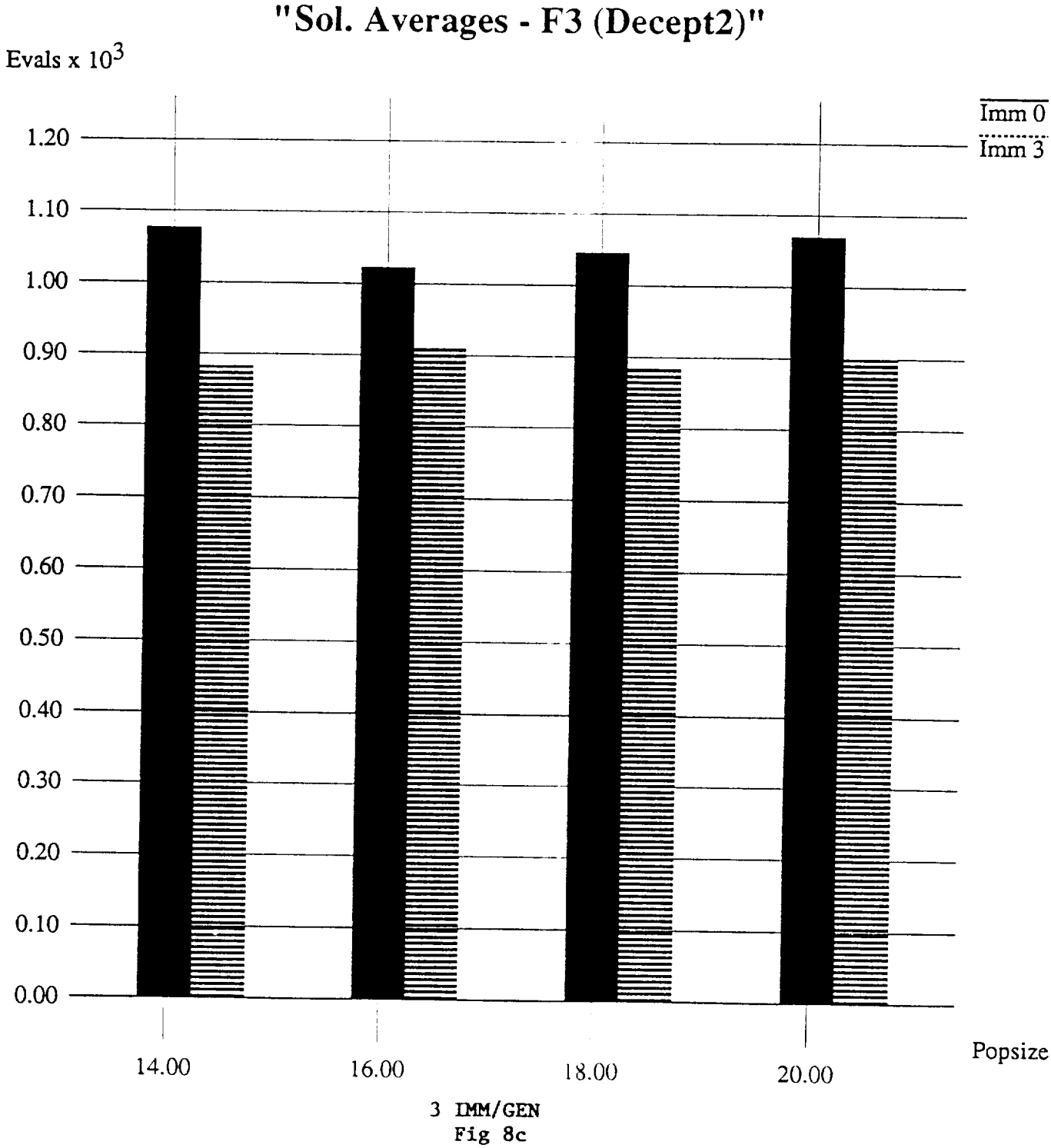
Fig 7b



2 IMM/GEN
Fig 8a



3 IMM/GEN
Fig 8b



3 IMM/GEN
Fig 8c

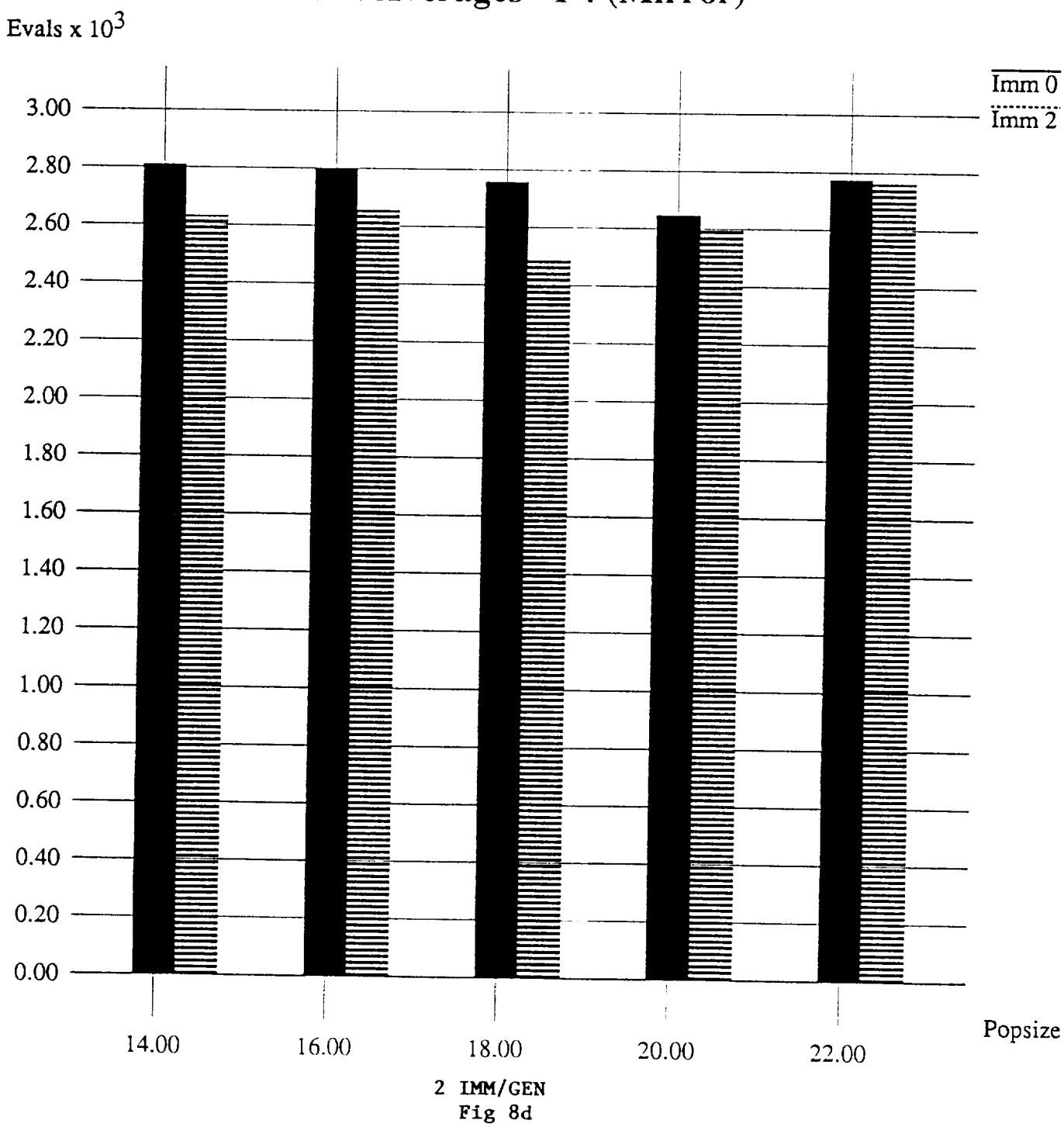

"Sol. Averages - F4 (Mirror)"
Evals x 10^3
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
Imm 0
Imm 2
14.00
16.00
18.00
20.00
22.00
Popsize
2 IMM/GEN

Fig 8d

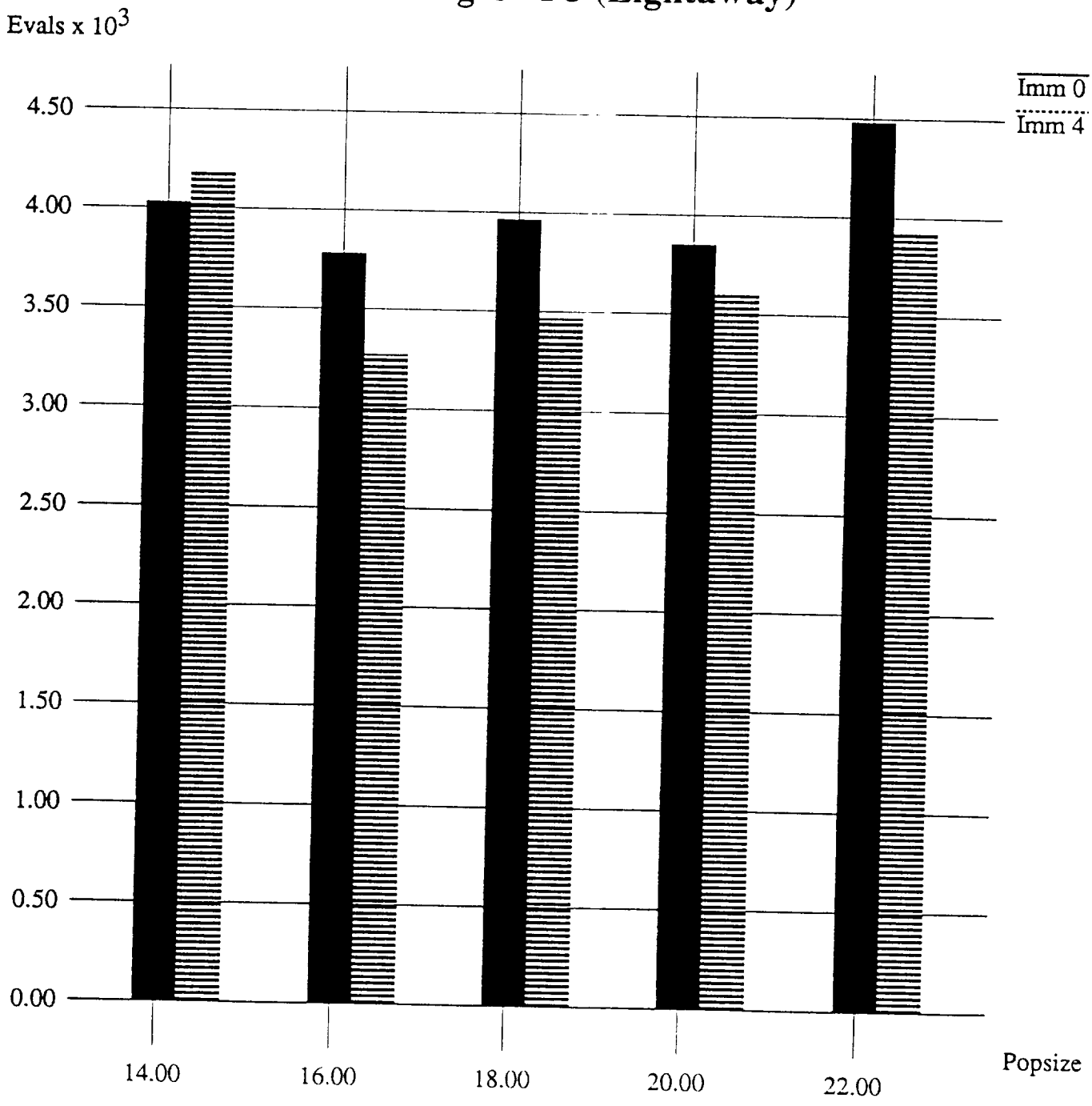

"Sol. Averages - F5 (Eightaway)"
Evals x 10^3
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
Imm 0
Imm 4
14.00
16.00
18.00
20.00
22.00
Popsize
4 IMM/GEN

Fig 8e